DIPLOMARBEIT

Web-basierte Datenbank zur Verwaltung administrativer Daten eines Universitätsinstituts

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Diplom-Ingenieurs
unter der Leitung von

Ao.Univ.Prof. Dipl.Ing. Dr.techn. Karl Riedling

E366

Institut für Industrielle Elektronik und Materialwissenschaften

eingereicht an der Technischen Universität Wien

Fakultät für Elektrotechnik

von

Harald Kleiner
Matr. Nr. 9625163
Franz Josef Straße 77
2344 Maria Enzersdorf

Wien, im September 2002

Kurzfassung

Universitätsinstitute werden heute in zunehmendem Maße wie Firmen geführt und autonom verwaltet. Damit kommen völlig neue Aufgaben auf die Verwaltung zu, die sich auf verschiedenste Bereiche erstrecken. Ob Personalstand, ausgeführte Projekte, zugeteilte Räume oder Beschäftigungsverhältnisse — alle diese Datenbestände weisen hohe Änderungsraten auf und müssen trotzdem stets konsistent vorliegen — mehr noch: sie sollen sogar in die Vergangenheit zurückverfolgt werden können, um vergangene Zustände rekonstruieren zu können.

Um die neu gestellten Anforderungen bewältigen zu können, bedarf es einer zuverlässigen Datenbank, die die Datenbestände zur Verfügung stellt und die die Benutzer bestmöglich unterstützt, die oft komplexen administrativen Vorgänge zu bewältigen. Dabei kommt natürlich der Datenbankoberfläche besondere Bedeutung zu. Sie soll den Benutzer bei seiner Arbeit unterstützen, Orientierungshilfen bieten, aber auch Eingabefehler erkennen und die nötige Arbeitsdisziplin einfordern, indem sie etwa verlangt, daß alle notwendigen Felder korrekt ausgefüllt werden.

Da die Umstellung der Administrationssoftware in jedem Falle enorme Anstrengungen erfordert, muß besonderer Wert auf eine dauerhafte Lösung gelegt werden. Insbesondere darf die Lösung nicht auf eine ganz bestimmte Softwareversion angewiesen sein, da dies eine Abhängigkeit vom Softwarehersteller bedeuten würde. Daher werden durchwegs offengelegte Standards und freie (im Sinne von “freie Entwicklung”) Software verwendet.

Als Ergebnis der Arbeit ist eine Web-basierte Datenbankanwendung entstanden, die fast vollständig plattformunabhängig ist. Am Server wird die Scriptsprache PHP eingesetzt, die seit Version 4 das objektorientierte Programmieren besonders unterstützt und fördert. Das Kernstück der Arbeit bestand im Aufbau von Klassen, die die Erstellung von Internetseiten mehr und mehr abstrahieren und in tiefer liegende Softwareschichten verlagern. Dadurch bleibt dem Programmierer die Sicht frei auf die eigentliche Anwendung; er muß sich nicht mehr direkt mit html-Programmierung auseinander setzen. Erst durch die konsequente Wiederverwendung von verallgemeinerten Programmteilen und systematische Einteilung in Klassen wird es überhaupt möglich, solch eine umfangreiche und komplexe Verwaltungssoftware ins Web zu bringen.

Summary

Institutes of European universities are more and more released from government property and led out into private enterprise management. Therefore new challenges have to be taken to manage various areas of administration. Just think of the people working for the institute, currently running projects or occupation of rooms – all these data have to be held ready for access even though they are changing rapidly – more than that: there is a strong demand to reconstruct states of information lying in the past.

To meet the new challenges a relyable database is required that provides users with data access and does its best to support users to manage even complex administrative procedures. Special importance comes to the database surface that has to support users in finding their way to achieve fast solutions but also to recognize wrong or untrustworthy input and thereby demand the necessary discipline on data entry.

Obviously the changing of database management system is a difficult task that takes a lot of effort. Therefore a long-term solution has to be achieved. The solution in particular must not rely on special releases of software packages to escape from dependencies of software developing companies. That is why open standards are employed and free software (i.e. development of software is free) is used entirely.

The result is a web-based database application which is almost completely platform-independent. PHP is used as serverside programming language; release 4 of PHP encourages object oriented programming. The main part of this work was to implement classes that abstract the task of html-page creation by moving code that creates html forms down into base classes. Following the concepts of modularization allows the programmer to concentrate on application programming without having to worry about all the html details. Only consequent re-use of generalized code segments allows to create a web based application for administration gaining the desired level of complexity and comfort.

Danksagung

Vielen Dank an Professor Riedling, der nicht müde wurde, meine vielen E-Mails innerhalb kürzester Zeit zu beantworten und der auch im Urlaub ein offenes Ohr für meine Fragen hatte. Es war für mich eine sehr große Unterstützung, auch auf so manche technische Frage neue Anregungen zu bekommen.

Mein ganz besonderes Dankeschön geht an meine Familie. Meine Eltern sind immer für mich da und geben mir die Ruhe und Sicherheit, um mich ganz auf die Arbeit konzentrieren zu können. Danke für die Unterstützung, das ist Euer Erfolg!

Melanie hat tapfer durchgehalten und die gesamte Arbeit korrekturgelesen – keine leichte Aufgabe bei den komplizierten Formulierungen.

Über diese Ausarbeitung

Diese Ausarbeitung wurde in LaTeX erstellt. Dabei hat die ‘doch nicht so kurze’ Einführung in LaTeX 2ε [17] sehr gute Dienste geleistet.

Die inhaltliche Strukturierung und Planung wurde unter intensiver Zuhilfenahme des Merkblattes für den Aufbau wissenschaftlicher Arbeiten von Karl M. Göschka [16] vorgenommen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Um die Institutsverwaltung zu vereinfachen und zu vereinheitlichen, soll ein Datenbanksystem entworfen werden, das die Datenhaltung für möglichst viele Bereiche integriert. Die gewünschten Anwendungsmöglichkeiten sind dabei so vielfältig, daß eigentlich bereits von einer “Verwaltungssoftware” gesprochen werden kann. Wenn auch das Ziel einer “Komplettlösung” nicht erreicht werden kann, so soll doch ein möglichst umfassender Themenbereich erfaßt werden.

Am Institut wurden bereits Erfahrungen mit einer einfacheren Verwaltungsdatenbank gemacht, die sich jedoch aus verschiedenen Gründen nicht bewährt hatte. Zum einen war die Funktionalität nicht ausreichend umfassend, es mußten also verschiedene uneinheitliche Werkzeuge parallel verwendet werden. Zum anderen bedurfte die bestehende Datenbank durch mangelnde Konsistenzerhaltung einer aufwendigen Wartung durch den Administrator, um die Anwendung funktionsfähig zu halten. Daher entstand der Wunsch nach einer möglichst umfassenden, betriebssicheren, dauerhaften und vor allem auch anwenderfreundlichen Lösung.

Ziel ist es nun, ausgehend von der bestehenden Datenbank das Datenmodell zu überarbeiten und zu erweitern, so daß es den erweiterten Aufgaben entspricht (‘umfassend’). Bei der Erweiterung soll bewußt Platz für zukünftige, noch nicht vorhersehbare Anforderungen eingeplant werden (‘dauerhaft’). Das Datenmodell wird auf einem Datenbankserver implementiert und als Grundlage für die zu schreibende Web-Anwendung benützt. Dadurch wird auch eine Trennung von Programm und Daten erreicht, unterstützt durch eine stärkere Modularisierung (‘betriebssicher’). Die Web-Anwendung muß die Datenbasis benutzergerecht präsentieren und dabei einen Kompromiß zwischen Komfort und Sicherheit finden. Beim Design der Oberfläche muß besondere Rücksicht auf Benutzer-

Orientierung genommen werden, damit die Anwendung auch von den Benutzern akzeptiert wird (‘anwenderfreundlich’).

Die weitere Ausarbeitung teilt sich in drei große Teile: Im folgenden Kapitel 2 wird zunächst einmal die Aufgabenstellung weiter präzisiert, um danach überhaupt zielgerichtet nach einer passenden Lösung suchen zu können. Das zweite Kapitel endet mit einem kurzen Einblick in die gewählte Systemumgebung, der zeigen soll, welch komplexe Vorgänge nötig sind, um einen Datenbankzugriff über das Internet vornehmen zu können.

Nachdem dann die Umgebung bekannt ist, soll der eingeschlagene Entwicklungsweg dargestellt werden. Dazu bietet das dritte Kapitel Gelegenheit. Zuerst wird als Basis das Datenbankmodell erstellt, um anschließend darauf aufbauend die gesamte Anwendung zu errichten. Dieser Teil spaltet sich noch in Teilbereiche auf, die beim Entwurf derart komplexer Software benötigt werden. Das dritte Kapitel endet mit einer Zusammenfassung darüber, welche der gesteckten Ziele erreicht wurden und was in weiteren Arbeitsschritten noch zu tun bleibt.

Das vierte Kapitel beinhaltet schließlich das Benutzerhandbuch für die entstandene Software. Obwohl weitgehend versucht wurde, die Software selbsterklärend zu gestalten, ist es doch im Sinne der Benutzer-Orientierung nötig, ein Nachschlagewerk beizufügen, das einerseits bei den ersten Schritten behilflich ist und andererseits auch den erfahrenen Benutzern noch Hilfestellungen für effizientes Arbeiten geben kann.

Kapitel 2

Problemstellung und Lösungsansatz

In diesem Kapitel soll zuerst einmal ausformuliert werden, unter welchen Rahmenbedingungen die Entwicklung durchzuführen ist und welche Anforderungen gestellt werden. Ausgestattet mit diesen Angaben wird anschließend nach einer Softwareumgebung gesucht, mit der die gesteckten Ziele am besten zu erreichen sind. Den Abschluß des Kapitels bildet eine Analyse der technischen Vorgänge in der gewählten Systemumgebung.

2.1 Aufgabenstellung und Ziele

Zur besseren Übersicht sind die Anforderungen in passende Themengebiete eingeteilt.

Datenbasis

Unter Datenbasis wird in diesem Zusammenhang der ‘Informations-Raum’ verstanden, der vom zugrunde liegenden ER-Diagramm aufgespannt wird – also die definierten Tabellen samt Attributen und Beziehungen. Das ER-Diagramm ist zwar erst in der Anfangsphase der Entwicklung entstanden (es lag also nicht bereits zu Beginn des Projektes vor), es wird aber nun zum Bestandteil der Aufgabenstellung erklärt[1].

[1] Die Erstellung des ER-Diagramms und seine Überprüfung im Interview mit dem Auftraggeber ist eigentlich eine detaillierte Formulierung der Aufgabenstellung, die der Entwickler für den Auftraggeber vornimmt. Die Interviews zur Erstellung des ER-Diagramms helfen sowohl dem Auftraggeber, seine Wünsche zu präzisieren, als auch dem Entwickler, um in die vom Auftraggeber verwendete Begriffswelt hineinzufinden.

Die Institutsdatenbank soll zunächst einmal folgende Entitäten (‘Dinge der realen oder der Vorstellungswelt’) verwalten: Personen, deren Tätigkeiten, Projekte, Institutstelefonklappen, Institutsschlüssel, dem Institut zugeteilte Räume, Computer und Netzwerkadressen sowie die hierarchische Struktur des Instituts. Erst aus der Art der zu speichernden Informationen und unter Beachtung der verwendeten Entwurfsregeln ergibt sich, daß noch einige weitere Entitäten berücksichtigt werden müssen, um die Information eindeutig rekonstruierbar abspeichern zu können. Diese zusätzlichen Entitäten stehen in engen Beziehungen zu den oben genannten, da sie eigentlich nur zusammengesetzte Attribute der zu speichernden Entitäten sind.

Da keine verbale Beschreibung so exakt sein kann wie das ER-Diagramm, wird an dieser Stelle das Diagramm selbst als Grundlage der Aufgabenstellung verwendet. (Siehe Bild 2.1).

Alle dargestellten Entitäten samt ihren Beziehungen müssen in der Datenbank gespeichert werden und durch die Oberfläche zugänglich gemacht werden. Für die Benutzbarkeit der zu erstellenden Formulare werden im folgenden Abschnitt Vorgaben getroffen.

Benutzerorientierung

Um den Benutzern den Umstieg auf die neue Anwendung zu erleichtern, sind mehrere Anforderungen zu beachten: Die Software muß also “aus der Sicht des Anwenders” entwickelt werden. Die am häufigsten vorkommenden Vorgänge sollten daher auch am besten unterstützt und automatisiert werden.

Mentalität: Die Software muß an den Benutzer angepaßt sein. Sie muß die Arbeit erleichtern, indem sie beispielsweise die Suche nach Datensätzen über geeignete Schlüsselwörter an den richtigen Stellen ermöglicht. Sie hat aber auch für die Korrektheit und Konsistenz der Daten zu sorgen, indem etwa Anomalien erkannt und verhindert werden oder offensichtlich falsch eingegebene Daten zurückgewiesen werden. Die Software muß also ‘geben’, damit sie von den Nutzern akzeptiert wird und ‘nehmen’, damit der Datenbestand stets aktuelle, vollständige und konsistente Daten enthält.

Hilfestellung: Grundsätzlich soll die Software möglichst selbsterklärend gestaltet werden, damit der Benutzer möglichst ohne weitere Hilfestellungen auskommt. Damit aber auch weniger geübte Benutzer leichter den Einstieg in die neue Soft-

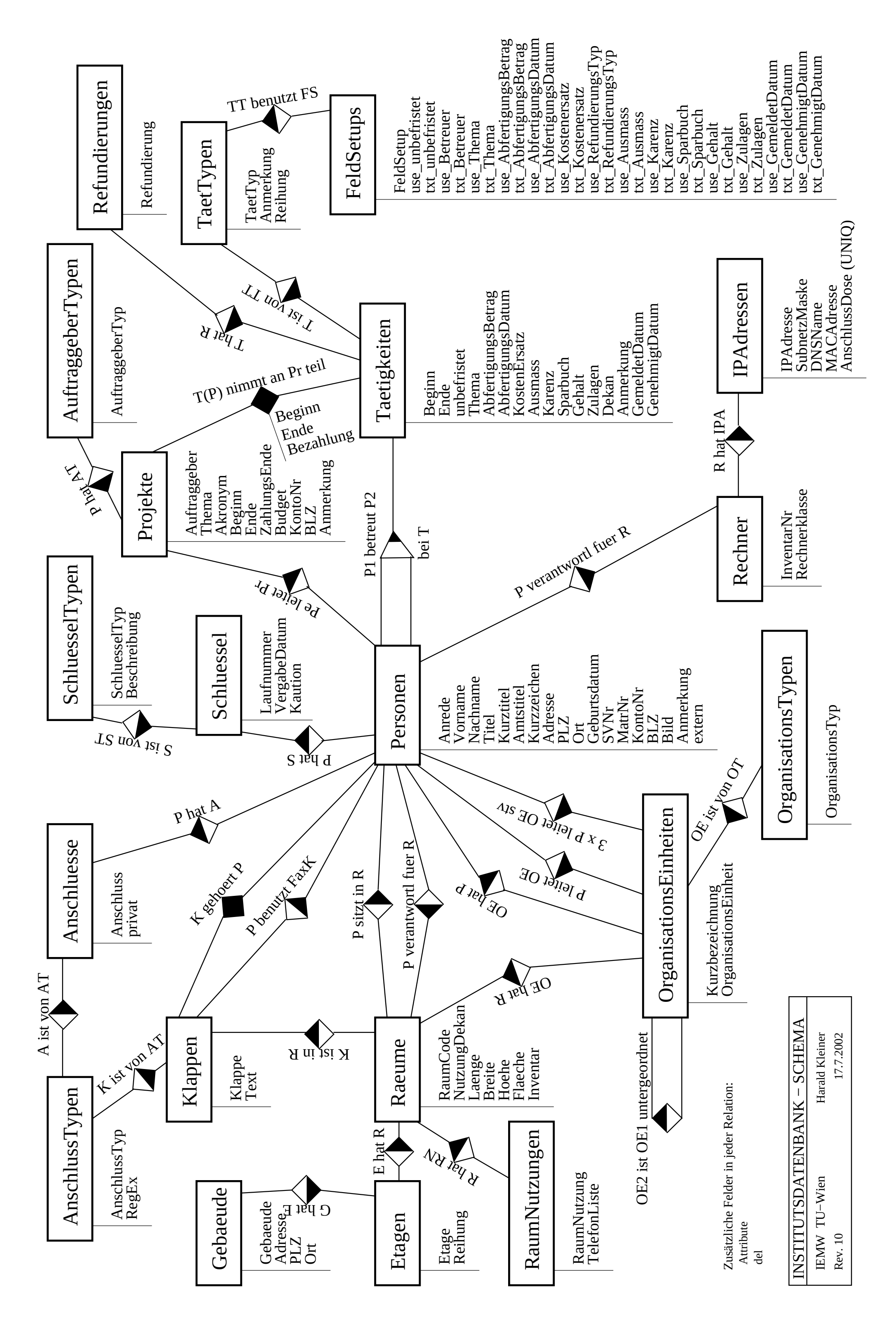


Abbildung 2.1: ER-Diagramm

ware finden, soll ein Hilfesystem integriert werden, das kontextabhängig gezielte Informationen bietet. Das Hilfesystem muß dazu online (d.h. als Bestandteil der Software) zur Verfügung stehen.

Navigation: Der Benutzer muß stets die Orientierung behalten können. Durch geeignete Navigationshilfen muß jederzeit erkennbar sein, in welchem Formular er sich gerade befindet. Auch der Wechsel zwischen Formularen muß dem Benutzer auf einfache Weise ermöglicht werden. In Anbetracht der vielen Verknüpfungsbeziehungen der Daten untereinander sind besonders effiziente Wege zu finden, um die Beziehungen anschaulich und übersichtlich darzustellen.

Optische Gestaltung: Selbstverständlich muß die Software ein ansprechendes Äußeres besitzen, ohne jedoch den Benutzer mit zu vielen Details oder unnötigen Design-Elementen zu verwirren. Eine einfache Möglichkeit, die Software auch nach Fertigstellung noch anzupassen wäre wünschenswert, da so auf Rückmeldungen der Benutzer in der Anfangsphase der Einführung reagiert werden kann.

Server

Als Server steht eine Maschine mit einem 500 MHz-Prozessor zur Verfügung. Es ist darauf zu achten, die vorhandenen Ressourcen effizient einzusetzen und akzeptable Antwortzeiten einzuhalten.

Client

Im Gegensatz zur bisherigen Lösung sollen möglichst viele Client-Plattformen die Anwendung benützen können. Es soll eine möglichst weitgehende Unabhängigkeit von Softwareversionen erreicht werden.

Da Benutzer sich im Allgemeinen von einer Anwendung beim Verlassen nicht abmelden, ist darauf zu achten, daß am Server keine Ressourcen auf Dauer belegt werden. Nach Ablauf von Sperrzeiten sind belegte und nicht mehr benutzte Ressourcen wieder freizugeben.

Archiv-Funktion

Eine wesentliche Anforderung an das System ist die Möglichkeit, auf Datenbestände aus der Vergangenheit zugreifen zu können. Daraus folgt unmittelbar, daß beim Löschen von Datensätzen diese nur als gelöscht markiert werden dürfen,

um die Daten nicht zu verlieren. Für ausreichend privilegierte Benutzer muß es anschließend möglich sein, die gelöschten Datensätze einzusehen und sie gegebenenfalls auch wiederherzustellen, falls sie irrtümlich gelöscht wurden.

Als Beispiel seien hier Beschäftigungsverhältnisse von Personen angeführt. Personen üben Tätigkeiten aus, unterbrechen diese möglicherweise und kehren in späterer Folge wieder ans Institut zurück. Beim Austritt der Person wird das Tätigkeitsverhältnis abgeschlossen und gelöscht. Kehrt die Person später wieder zurück, so können die gelöschten Tätigkeiten als *Vorlage* zur Eingabe der neuen Tätigkeit dienen, sie dürfen aber keinesfalls wiederhergestellt und 'weiterverwendet' werden, da ja eine neue Tätigkeit begonnen wird. Durch eine Weiterverwendung würde die in der Vergangenheit liegende Tätigkeit überschrieben und wäre verloren. Wurde die Person inzwischen gelöscht, darf – und soll – der Personen-Datensatz wiederhergestellt werden, da es sich ja um dieselbe Person handelt.

Zugriffskontrolle

Die Datenbank enthält sensible, teilweise sogar geheimhaltungspflichtige Daten. Diese Daten müssen unbedingt vor unbefugtem Lesezugriff geschützt werden. Weiters darf auch ein Schreibzugriff nur bestimmten, von der Institutsleitung zu bestimmenden, Benutzern gestattet werden. Darüber hinaus hat jeder Benutzer seinen speziellen Aufgabenbereich, der sich von anderen Bereichen stark abgrenzt. Beispielsweise sollen Benutzer auf ausgewählte Daten Schreibzugriffsrechte bekommen, gleichzeitig aber andere Datenbereiche nicht einmal lesen dürfen.

Als Konsequenz aus diesen Anforderungen muß ein Privilegiensystem realisiert werden, das es ermöglicht, jedem Benutzer explizit Berechtigungen für bestimmte Bereiche der Datenbank zu gewähren. Es muß dabei bis auf die Feld-Ebene herunter (also für jede einzelne Spalte einer Tabelle) möglich sein, Privilegien für bestimmte Benutzer zu vergeben.

Trotz dieses hohen Detaillierungsgrades muß jedoch eine einfache Administrierbarkeit für das Privilegiensystem gewährleistet sein. Da meist mehrere Benutzer gleichartige Berechtigungsprofile zugesprochen bekommen, wird eine Einteilung in Gruppen zweckmäßig sein. Diese Einteilung muß aber flexibel sein, d.h. jeder Benutzer muß in mehreren Gruppen Mitglied sein können und es muß jeweils das stärkste Recht aus allen Gruppen zur Bewertung herangezogen werden. Aus Gründen der Transparenz ist es nötig, die tatsächlichen Berechtigungen eines Benutzers ermitteln zu können, um die eingestellten Gruppenzugehörigkeiten auf

Korrektheit überprüfen zu können.

Umstieg von der bestehenden Datenbank

Datenexport: Der derzeitig existierende Datenbestand muß vollständig und selbstverständlich unter korrekter Beibehaltung der Beziehungen zwischen den Datensätzen in das neue System importiert werden. Dazu ist ein Datenbank-Frontend vorzusehen, das an den Datenbestand ankoppelt und die vorhandenen Daten für den Import in die neue Datenbank aufbereitet.

Installation: Da die Konsistenz-Prüfungen in der bestehenden Datenbank nicht ausreichend streng formuliert sind, wird es nötig sein, noch vor der tatsächlichen Inbetriebnahme manuell Korrekturen am Datenbestand vorzunehmen, so daß die gewünschten höheren Qualitätsstandards eingehalten werden. Anschließend sind die Prüfungsregeln entsprechend zu verschärfen, damit die so erreichte Qualität auch erhalten bleibt.

Inbetriebnahme: Die Umstellung auf das neue System hat möglichst ohne Betriebsunterbrechung zu erfolgen; gleichzeitig muß ab Umstellungstermin sichergestellt sein, daß die ‘alte’ Datenbank anschließend nicht mehr zugänglich ist.

2.2 Lösungsansatz

Auf der Suche nach einer Web-fähigen Alternative zu Microsoft Access sollen nur Softwarepakete berücksichtigt werden, die aufgrund ihrer langen Entwicklungszeit und ihrer weiten Verbreitung als stabil, erprobt und dauerhaft bezeichnet werden können. Selbstverständlich spielen auch finanzielle Aspekte wie Lizenzgebühren und Kosten für Support im Umfeld eines Universitätsinstituts eine besonders wichtige Rolle. Außerdem soll eine Bindung an bestimmte Softwarehersteller möglichst vermieden werden. In den folgenden Absätzen wird die Auswahl der beteiligten Softwarekomponenten kurz erläutert und begründet.

Betriebssystem: Das Betriebssystem als Fundament der gesamten Serverarchitektur muß vor allem eine stabile, sichere und leistungsfähige Plattform bieten. Beim Server kann gut auf graphische Benutzeroberflächen verzichtet werden, für die Clients stehen sie ohnehin zur Verfügung.

Die Auswahl beschränkt sich aufgrund des vorhandenen Know-hows und der vorhandenen Hardware auf Windows sowie UNIX und Linux. Die beiden Letztgenannten bieten vergleichbaren Leistungsumfang, jedoch ist Linux frei verfügbar während UNIX-Derivate käuflich zu erwerben sind.

Von den am Markt befindlichen Windows-Versionen sind nur die Server-Versionen (Windows 2000 oder XP) geeignet; die Desktop-Versionen gelten als nicht ausreichend stabil. Gegen den Einsatz von Windows als Serverbetriebssystem sprechen mehrere prinzipbedingte Schwächen, die bei Linux nicht zu finden sind: Wählt man Windows als Betriebssystem, so beschränkt sich die Auswahl der nativ für dieses Betriebssystem programmierten Webserver neben einigen 'kleineren' Produkten auf den Internet Information Server (IIS) von Microsoft. Die meisten anderen Webserver stammen aus der UNIX/Linux-Welt und wurden auf Windows portiert, was Einbußen in Bezug auf Leistungsfähigkeit und Sicherheit zur Folge haben kann. Weiters gehen neuere Versionen von Windows verschwenderisch mit Ressourcen um, während Webserver auf Linux-Basis im Textmodus selbst auf Computern der 80486-Klasse noch funktionieren.

Um den Beschränkungen der vorhandenen Hardware und des Budgets gerecht zu werden, fällt die Wahl klar auf Linux als Serverbetriebssystem.

Web-Server: Die wichtigsten Anforderungen an den Webserver beziehen sich auf Geschwindigkeit und Sicherheit gegenüber Angriffen von außen. Außerdem muß der Webserver mit der verwendeten Programmiersprache zusammenarbeiten können. Optimal wäre eine Kombination, bei der die Programmiersprache modulartig in den Server integriert ist. Dies vermeidet die Leistungseinbußen, die CGI-Systeme durch die Eröffnung eines eigenen Prozesses zum Parsen des Seitenquelltextes haben.

Aus Gründen der Sicherheit kommt der Internet Information Server keinesfalls in Frage.

Für Windows existieren noch weitere 'kleinere' Webserver, wie beispielsweise der Sambar-Server. Dieser ist zwar kostenlos erhältlich, im Unterschied zu *freier* Software sind die Quelltexte allerdings nicht verfügbar. Das bedeutet eine Bindung an einen (in diesem Falle noch dazu besonders kleinen) Softwarehersteller, die ja vermieden werden sollte.

Da als Betriebssystem Linux verwendet wird und keine zwingenden Argumente gegen den unangefochtenen Marktführer Apache sprechen, wird Apache als Server eingesetzt. Er bietet äußerst flexible Konfigurationsmöglichkeiten, Modul-Schnittstellen zu den meisten Programmiersprachen und ist aufgrund sei-

ner langen und aktiven Weiterentwicklung als besonders ausgereift zu bezeichnen.

Programmiersprache: Die zu verwendende Programmiersprache soll die Programmierung von Datenbankzugriffen effizient unterstützen, die geschriebenen Programme sollen ressourcenschonend und performant ablaufen und leicht wartbar sein. Dazu soll die Sprache modulares (beispielsweise objektorientiertes) Programmieren unterstützen. Die Sprache soll selbstverständlich gut dokumentiert und leicht erlernbar sein, um einen raschen Projektfortschritt zu ermöglichen.

Bei den Programmiersprachen bietet sich die größte Auswahl, da immer wieder neue Sprachen am Markt auftauchen. Es muß daher besonders sorgfältig auf Eignung und dauerhafte Unterstützung geachtet werden.

Server Side Includes (SSI) sind Tags, die in den normalen html-Quelltext eingebettet werden und vom Webserver selbst ausgewertet werden, bevor die Seite ausgeliefert wird. Mit SSI können jedoch nur einfache Konstrukte realisiert werden; von einer echten Programmiersprache kann kaum gesprochen werden.

Java bietet mit Java Server Pages (JSP) und Zugriff auf die Java Enterprise Edition eine besonders professionelle Entwicklungsumgebung, die auch für größere Projekte geeignet sein dürfte. Der Einarbeitungsaufwand dürfte jedoch für das vorliegende Projekt nebst den Anforderungen, die an die Serverhardware gestellt werden, zu groß sein.

ASP ist das Microsoft-Gegenstück zur CGI-Schnittstelle und bietet Unterstützung für verschiedene Sprachen – zumeist wird VBScript oder JavaScript verwendet. Diese Sprachen fördern jedoch nicht das ‘elegante’ Programmieren und gehören außerdem in die Windows-Welt von IIS.

PHP ist mit seiner C-ähnlichen Syntax und seiner vereinfachten Zeichenkettenverarbeitung wie in Pascal oder Java (in PHP muß sich der Programmierer nicht um Zeiger oder Reservierung von Speicher kümmern) sehr leicht zu erlernen und gleichzeitig einfach anzuwenden, da die meisten Sprachkonstrukte ohne Nebenbedingungen funktionieren. Weiters bietet die Sprache Objektorientierung und eine große Anzahl eingebauter Funktionen und APIs zu verschiedensten anderen Programmen.

So bleibt als Hauptkonkurrent zu PHP nur noch Perl, das als Zeichenkettenverarbeitungssprache begonnen hat und ebenfalls sehr erfolgreich am Markt ist. Da jedoch PHP leichter erlernbar ist und das Schreiben gut lesbarer Programme mehr fördert als Perl, wird PHP der Vorzug gegeben.

Datenbank: Um standardkonform zu sein, muß jedenfalls eine SQL-Datenbank verwendet werden. Die Natur der zu speichernden Daten verlangt außerdem eine Realisierung mittels einer relationalen Datenbank, wobei Transaktionen (Aneinanderreihung von Teil-Vorgängen, die im Fehlerfall vollständig zurückgenommen werden kann) nicht benötigt werden. Des weiteren kann auf die Forderung nach ausgefeilten SQL-Konstrukten verzichtet werden, um eine möglichst hohe Verarbeitungsgeschwindigkeit zu erzielen.

In die engere Auswahl kommen die freien Datenbanken MySQL und PostgreSQL sowie die kommerziellen Oracle und Microsoft Access.

Access wäre im Prinzip als Backend-Datenbank geeignet, scheidet jedoch aufgrund mangelnder Leistungsfähigkeit und der Bindung an Windows als Betriebssystem aus. Oracle erscheint aufgrund der zahlreichen Features zu schwerfällig für die zu verwendende Serverhardware und aufgrund der immensen Lizenzgebühren für nicht geeignet. PostgreSQL ist im Vergleich zu MySQL für komplexere Abfragen geeignet; in MySQL müssen dafür einfachere Konstrukte mehrfach kombiniert werden. Da sich die benötigte Komplexität in Grenzen hält und großer Wert auf eine ausführliche Dokumentation und weite Verbreitung gelegt wird, wird MySQL als Datenbankserver gewählt [2].

2.3 Eigenschaften von LAMP

Die Beschreibung von Linux, Apache, MySQL und PHP würde einige Bücher füllen. Daher soll hier nur auf ausgewählte Aspekte eingegangen werden, die besondere Auswirkungen auf die Programmierung von Internet-Applikationen haben.

Eine wesentliche Eigenschaft von http ist, daß aufeinanderfolgende Zugriffe keinerlei Beziehung zueinander aufweisen – das Protokoll ist *zustandslos* (stateless). Das bedeutet, daß die Zugriffe der Benutzer durch die übergebenen Parameter identifiziert und voneinander getrennt werden müssen. Dies wird durch die sogenannte SessionID bewerkstelligt. Es handelt sich dabei um eine Zeichenfolge, die zu Beginn einer Sitzung vergeben wird und anschließend in jede abgerufene Seite aufgenommen wird. Bei einem eintreffenden Seitenaufruf werden mit Hilfe der SessionID die Zustände der Anwendung (also Variablen in der verwendeten Programmiersprache) aus der Datenbank wiederhergestellt. Beinhaltet ein Seitenaufruf keine SessionID, so wird einfach eine neue ID vergeben und eine neue

Sitzung eröffnet.

Da die Applikation über den Webserver wie jede andere Internetsite erreichbar ist, sind besondere Vorkehrungen gegen unerwünschte Eindringlinge vorzusehen. Die sicherste Betriebsart wäre, den Server in einem privaten Subnetz zu betreiben, so daß er von außerhalb der TU Wien nicht erreicht werden kann. Wird Zugriff von außen gefordert, so muß die Server-Konfiguration besonders sicher ausgelegt sein. Der Serverprozeß darf dann nur die Privilegien besitzen, die unbedingt nötig sind, um die Seiten ausliefern zu können. Selbstverständlich muß *vor* Inbetriebnahme des Servers die Konfigurationsdatei Punkt für Punkt auf die gewünschten Sicherheitseinstellungen überprüft werden. Weiters wird der Einsatz verschlüsselter Datenübertragung (SSL) erwogen, um sowohl die Zugangsdaten als auch die von der Datenbank abgerufenen Nutzdaten gegen Abhören zu sichern.

Um eine Web-Applikation robust zu gestalten, dürfen Benutzereingaben niemals ungeprüft übernommen werden. (Dies gilt für Datenbankanwendungen generell – für Web-Applikationen jedoch ganz besonders, da im allgemeinen die Anwender einer Web-Applikation entweder völlig unbekannt sind oder aus psychologischen Gründen wenig Arbeitsdisziplin zeigen, da sie sich ‘unbeobachtet’ fühlen.) Diese Prüfungen verfolgen zwei Ziele:

1. Erhaltung der Datenintegrität: Die Anwendung muß erkennen, wenn Benutzer unsinnige oder zu kurze Daten eingeben. Zu diesem Zweck wird für jedes unbedingt auszufüllende Feld eine Mindestlänge (in Zeichen) vorgegeben. Außerdem wird entsprechend dem vorgegebenen Datentyp jedes Feldes eine Überprüfung mittels regulärer Ausdrücke vorgenommen, um die Plausibilität der eingegebenen Daten sicherzustellen.

2. Schutz vor Angriffen von außen: Eine einfache aber durchaus wirkungsvolle Attacke gegen eine Web-Applikation ist es, in ein Textfeld Daten einzugeben, die vom Datenbankserver als Befehl interpretiert werden und so Schaden an den Daten anrichten. Würde beispielsweise ein Benutzer in ein Textfeld den Wert `Text"; DROP DATABASE Institut;` eingeben, so würde die Anweisung zum Abspeichern der Daten nach dem Wort ‘Text’ durch das eingegebene Anführungszeichen vorzeitig zu Ende sein und anschließend die `DROP DATABASE` - Anweisung ausgeführt werden. Dies kann jedoch zuverlässig verhindert werden,

indem Sonderzeichen (z.B. die Anführungszeichen) in allen Eingabefeldern maskiert (escaped) werden. Dadurch kann der String-Kontext in der SQL-Anweisung nicht vorzeitig verlassen werden.

2.3.1 Überblick und Literatur

Die Erstellung von Web-Applikationen ist eine multidisziplinäre Aufgabe. Wie noch gezeigt wird, kommunizieren mehrere Prozesse über unterschiedliche Protokolle und Sprachen miteinander. Der Programmierer hat dabei den Überblick über die tatsächlichen Vorgänge zu behalten und muß gleichzeitig in verschiedenen Sprachen denken und programmieren. An manchen Stellen ist es unvermeidlich, unterschiedliche Sprachen in derselben Datei zu verwenden oder sogar mit einer Sprache Konstrukte zu erstellen, die in einer anderen Sprache ausgeführt werden sollen (beispielsweise Erstellung eines php-Programms, das JavaScript-Code ausgibt, der Daten aus der Datenbank enthält)

html: In dieser Anwendung wird html als Kommunikationssprache mit dem Client (einem Internet-Browser) verwendet. Obwohl gängige Browser tolerant gegenüber Fehlern im html-Quelltext sind, muß trotzdem auf Vollständigkeit und Standardkonformität geachtet werden, damit die Anwendung in möglichst vielen Browsern korrekt dargestellt wird. In diesem Zusammenhang sei html-tidy, ein Validierungsprogramm für html zu empfehlen. Mit diesem Hilfsprogramm kann festgestellt werden, ob eine html-Datei den vom W3 Konsortium [11] empfohlenen Standards entspricht.

Literatur zu html existiert mittlerweile in unüberschaubarem Maße. Ganz besonders hervorzuheben ist jedoch SelfHTML von Stefan Münz [3] – nicht nur, weil es kostenlos erhältlich ist. Hier werden neben den Funktionen von html, Style Sheets, JavaScript und XML auch spezielle Hinweise zu häufigen Fehlerquellen und zur Web Usability gegeben.

JavaScript: Mittels JavaScript lassen sich Vorgänge realisieren, die direkt am Client-Computer durch die Browsersoftware ausgeführt werden. Da die von JavaScript ausgeführten Befehle keinen direkten Einfluß auf die Daten am Server nehmen können, eignet sich JavaScript hauptsächlich zur Beschleunigung von Aufgaben, die ebensogut vom Server ausgeführt werden könnten, nur eben einen zusätzlichen Neuaufbau der Seite erfordern würden oder zum Erzeugen von Bestätigungs-Dialogen, jedoch nicht für echte Datenmanipulationen.

Da JavaScript - wie html - vom Client ausgeführt wird, muß hier besonders auf Standardkonformität geachtet werden. Zusätzlich sind besondere Testläufe auf möglichst vielen Softwareversionen möglichst vieler unterschiedlicher Browser nötig.

Zu JavaScript existieren zwei Handbücher von Netscape [4], [5]. Es ist jedoch zu beachten, daß die von Microsoft implementierten JavaScript-Versionen nicht vollständig kompatibel zum Netscape-Standard sind.

php ist die zentrale Sprache. Der Webserver arbeitet php-Skripte ab, um als Ergebnis html-Quelltexte zu erhalten, die an den Client zurückgesendet werden. Dabei werden grundsätzlich Blöcke von php-Code in den html-Quelltext eingebaut. Die Schreibweise ist vergleichbar mit JavaScript-Code, es wird jedoch der php-Block bei Abruf der Seite durch das php-Modul ausgewertet und durch seine Ergebnisse ersetzt. Es ist auch erlaubt, daß eine Datei nur aus einem einzigen php-Block besteht bzw. daß kein php-Block existiert (eine html-Datei ist also immer auch ein gültiges php-Programm).

Zum ersten Einstieg in php ist das PHP-HOWTO [7] gut geeignet. Der Leser wird aber nach kurzem Studium desselben bald auf die offizielle PHP-Dokumentation umsteigen wollen [8], die besonders umfassend ist und alle php-Funktionen behandelt. Häufig gestellte Fragen beantwortet die Zusammenfassung der deutschsprachigen php-Newsgroup [9].

SQL: Mittels SQL wird auf die Datenbank zugegriffen. Obwohl die Sprache genormt ist, implementieren Datenbankhersteller jeweils proprietäre Erweiterungen, um Kunden an das eigene Produkt zu binden (‘an extended subset of SQL’). Um solch eine Bindung zu vermeiden, läßt sich MySQL im sogenannten ‘ANSI-Mode’ betreiben, in dem es nur genormte Konstrukte akzeptiert. Da MySQL auch auf lange Sicht für diese Anwendung optimal erscheint, wurde die ANSI-Konformität nicht getestet.

Als Literatur zu MySQL ist ebenfalls das originale Handbuch sehr zu empfehlen [6]. Es enthält neben der Sprachreferenz ein sehr lebendig gestaltetes SQL-Tutorial, das den Leser anhand von Beispielen in SQL einführt.

Usability: Die Benutzerorientierung hat in diesem Projekt eine besonders hohe Priorität. Deshalb soll an dieser Stelle auch ein Handbuch zur Web Usability erwähnt werden [10], das neben den Prinzipien der Usability auch ausführlich

Design-Regeln für gut geeignete Web-Präsenzen behandelt. Anhand vieler Beispiele wird auch auf Fehler bekannter Webseiten eingegangen.

2.3.2 Fallstudie: Ablauf einer http-Session

Hier soll demonstriert werden, daß der Abruf einer Webseite mit Datenbankanbindung zwar aus einer beträchtlichen Anzahl von Einzelschritten aufgebaut ist, diese Schritte sind jedoch – jeder für sich genommen – leicht zu verstehen. Der dargestellte Ablauf ist das Ergebnis konsequenter Modularisierung und Zerlegung von komplexen Aufgaben in kleinere, leichter lösbare Teil-Aufgaben. Erst durch die Modularisierung wird es möglich, hochkomplexe Systeme aufzubauen, die schließlich auf einer sehr hohen Abstraktionsebene miteinander kommunizieren. Das Ergebnis ist eine Aneinanderkettung solch einfacher Teil-Vorgänge die jedoch schon bei einfachen Gesamtvorgängen unübersichtlich wird, wenn man nicht den großen Zusammenhang vor Augen hat.

In den folgenden Absätzen werden einige Abkürzungen verwendet, die an dieser Stelle nicht genauer erklärt werden können, im Anhang A findet sich eine Auflistung mit kurzen Erklärungen. Als Nachschlagewerk kann hier auch das TCP-Tutorial von IBM [1] sehr empfohlen werden.

Im dargestellten Beispiel stellt ein Internet-Browser an einen Webserver eine Anfrage. Dieser wiederum besorgt sich die Daten für seine Antwort von einem Datenbankserver.

Um das Beispiel überschaubar zu halten, besteht die Antwort einfach nur aus dem Namen einer Person. Der Browser übergibt am Beginn die PersonID[2], um die gewünschte Person festzulegen.

In Abb. 2.2 ist die Ausgangslage zu sehen. Im linken Teil ist der Client (die Browsersoftware des Benutzers) und rechts der Server dargestellt. Ausgetauschte Nachrichten werden als Pfeile dargestellt. Der Client besitzt die IP-Adresse 10.100.17.50; der Server hat die Adresse 10.100.17.10 und wartet auf dem Standard-Port 80 auf eingehende Verbindungen.

Zu Beginn hat der Benutzer die gewünschte Adresse im Browser-Fenster eingegeben und schickt seine Anfrage nun ab. Der Browser beginnt nun, die tatsächli-

[2]Die PersonID ist einfach nur eine eindeutige Zahl, die jeder Person zugewiesen wird, vergleichbar mit einer Zeilennummer in einer Tabelle (PersonID ist das Primärschlüsselfeld der Personen-Tabelle)

che Netzwerkadresse des Servers (die MAC-Adresse) zu ermitteln, bevor er anschließend eine TCP-Session öffnet, um die Anfrage tatsächlich abzusetzen.

Der erste Schritt (1) ist also ein ARP-request. Der Browser stellt an alle lokal angeschlossenen Rechner die Frage (Ethernet Broadcast), wem die vom Benutzer eingegebene IP-Adresse gehört. Läge der Server in einem anderen Subnetz, so würde der Router mit seiner eigenen MAC-Adresse antworten und die kommenden Anfragen an den Server weiterleiten. In diesem Fall liegen aber Client und Server auf demselben Netz und so antwortet der Server selbst. Von nun an kann der Browser den Server direkt über das lokale Netzwerk ansprechen; er beginnt auch sofort, die TCP-Session zu eröffnen (3).

Eine TCP-Session wird im sogenannten three-way handshake eröffnet, indem die beteiligten Rechner sequence numbers austauschen und den Empfang jeweils bestätigen (Sequence numbers numerieren die übertragenen Datenpakete und werden in jeder Session auf zufällige Werte initialisiert, damit nicht verzögerte Pakete aus vorangegangenen Sessions die Übertragung durcheinander bringen können).

Der Client hat (zufällig) entschieden, Port 1032 als Eingangsport zu verwenden, und hat die neue sequence number mit 17 festgelegt. Er schickt nun ein TCP SYN-Paket (‘synchronize’), das diese Daten enthält, an den Server. Der Server antwortet mit einem analogen Paket (4), das die sequence number für seine Pakete enthält (124 in diesem Fall) und gleichzeitig den Empfang des Paketes (3) bestätigt. Nachdem der Browser den Empfang dieses Pakets bestätigt hat (5), besteht zwischen Browser und Server eine gültige TCP-Session: Zwei unabhängige Datenkanäle zwischen den Ports 1032 (Client) und 80 (Server) ermöglichen die Kommunikation in beide Richtungen.

Nun sind die Vorarbeiten abgeschlossen und der Client kann die gewünschte Anfrage absetzen. Er schickt dazu ein http-GET-Paket (6) an den Server. Dieses GET-Paket enthält neben weiteren Informationen über den Client den vom Benutzer angegebenen Link. An Zusatzinformation stehen im GET-Paket beispielsweise die Version des verwendeten Browsers (user agent), die bevorzugte Sprache und die Bildformate, die der Browser verarbeiten kann.

Nachdem der Server in (7) den Empfang des GET-Paketes bestätigt hat, endet hier das erste Drittel der Verarbeitung. Im zweiten Drittel bearbeitet der Server die Anfrage und stellt eine Antwort zusammen, im letzten Drittel wird die Antwort an den Browser zurückgeschickt und die TCP-Session abgebaut.

Bis jetzt hat der inetd (Internet Daemon) mit dem Client kommuniziert. Nach dem http-GET-Paket hat jedoch der inetd die Kontrolle an den httpd (der eigentli-

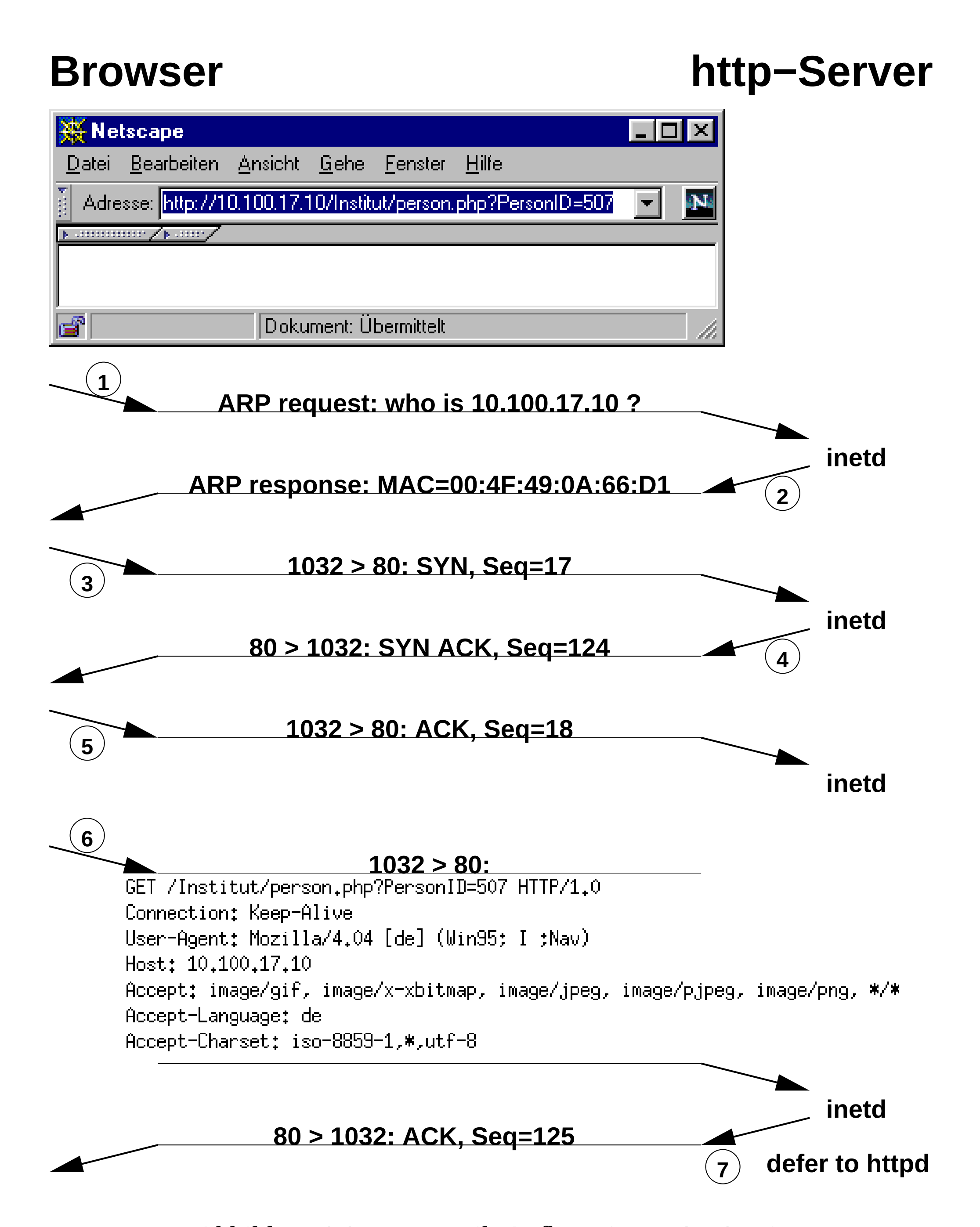


Abbildung 2.2: Erster Teil: Aufbau einer TCP-Session

che Webserver-Prozeß, hier Apache) übergeben. Der inetd dient so als Vermittler zwischen Protokollen, Ports und Server-Prozessen.

Im zweiten Teil – der Bearbeitung der GET-Anfrage – wechseln nun die Rollen. Während der Browser auf eine Antwort wartet, wird jetzt der httpd (der Web-Server) zum Client gegenüber dem Datenbankserver:

Nach Erhalt der GET-Anfrage übergibt der httpd die Bearbeitung der Anfrage an einen von mehreren wartenden Kind-Prozessen. Die Kind-Prozesse werden bereits beim Start des httpd erstellt (fork()[3]) und bearbeiten mehrere Anfragen, bevor sie aus Sicherheitsgründen beendet und wieder erneuert werden. Der Kind-Prozeß bekommt die Daten aus dem GET-Paket als Aufrufparameter übergeben und startet seinerseits das integrierte php-Modul, um die Quelldatei zu interpretieren.

Im Bild 2.3 ist oben der Quelltext der angeforderten Datei abgedruckt. Das Programm richtet eine Anfrage an die zugrundeliegende Datenbank und gibt das Ergebnis als minimalen html-Quelltext aus.

Im ersten Schritt erzeugt das Programm eine neue Instanz der Datenbank-Zugriffs-Klasse. Diese Klasse ist Bestandteil von PHPLIB und dient dem Zugriff auf MySQL-Datenbankserver. Die hier verwendete (abgeleitete) Klasse `DB_Institut` ist identisch mit der originalen Klasse, es wurden nur in der abgeleiteten Version die Zugangsdaten (Server-Adresse, Benutzername und Paßwort) des Datenbankservers überschrieben. Im zweiten Schritt wird an die Datenbankklasse eine SQL-Abfrage übergeben. Die Abfrage holt aus der Personen-Tabelle jene Vor- und Nachnamen, bei denen das Primärschlüsselfeld (PersonID) übereinstimmt. Aus Sicherheitsgründen wird die Variable `PersonID` durch die Funktion `addslashes()` 'vorbehandelt', damit eventuell vorhandene Sonderzeichen in der SQL-Anweisung keine Schäden anrichten können.

Der Einfachheit halber sei angenommen, daß Webserver und Datenbankserver auf demselben Computer laufen. Dadurch erfolgt die Kommunikation zwischen den Beiden über Unix Sockets (also nicht über TCP und das Netzwerk) und wir beschränken uns hier auf die wesentlichen Schritte der Kommunikation.

Die Methode query() erkennt nun, daß noch keine Verbindung zum Datenbankserver besteht und baut diese sofort auf (8). Der mysqld (MySQL-Daemon, der Datenbank-Server-Prozeß) ist multithreaded aufgebaut; es wird also ein neuer Thread erzeugt, der die Anfrage entgegennimmt. Als Antwort meldet der mys-

[3]fork() erzeugt einen absolut identischen, geklonten Prozeß. Um diesen aufwendigen Vorgang zu beschleunigen, wird die angekündigte Version 2.0 des Apache-Webservers Multithreading unterstützen.

qld eine Identifikation für die Session (handle) an den Client (den Web-Server-Prozeß!) zurück (9). Dieser übermittelt anschließend seine SQL-Abfrage (10).

Der mysqld überprüft die Abfrage auf Fehler und führt sie aus, wenn sie syntaktisch richtig ist. Nach erfolgreicher Ausführung der Abfrage hält der Datenbankserver-Prozeß die Ergebnisdaten bereit zur Abholung durch den Client – sie werden jedoch noch nicht übertragen. Der Client bekommt vorerst nur eine Statusmeldung (11), die das Vorhandensein der Daten anzeigt. Anschließend kann der Client selbst entscheiden, wann er die Ergebnisse abholen möchte. Mit jedem Aufruf von `next_record()` (12) bekommt der Client eine Zeile aus der Ergebnismenge retourniert (13), bis keine Daten mehr anstehen. In diesem einfachen Beispiel existiert nur eine einzige Zeile; in der Praxis wird aber meist eine Schleifenstruktur verwendet, um alle Daten abzufragen.

Am Ende des PHP-Programms wird die Ausgabe gestaltet. Mit dem Befehl `echo` wird zuerst der Kopf der html-Seite ausgegeben. Anschließend werden Vor- und Nachname aus der Datenbankabfrage eingefügt und am Ende der html-Text mit den schließenden Tags vervollständigt.

Damit ist die zweite Phase abgeschlossen: Das PHP-Programm ist beendet, die Verbindung zum Datenbankserver (implizit) geschlossen worden und die gesuchte html-Seite steht zur Verfügung und kann an den Browser geschickt werden.

Nun startet die letzte Phase: Die Übertragung der html-Seite zum Browser und der Abbau der Verbindung. In dieser Phase sind die Rollen wieder wie zu Beginn: Der Browser ist Client und der httpd ist Server (Abb. 2.4).

Die letzte Aktion des httpd vor seiner Beendigung ist die Übertragung der html-Seite (14). Das übertragene Paket enthält den Statuscode ('200, OK'), das Ablaufdatum der Daten, Anweisungen zum Cachen der Daten und natürlich im Body des Pakets die html-Seite.

Nun muß nur noch die TCP-Session abgebaut werden. Dazu setzt der httpd ein "FIN, ACK"-Paket(15) (finish, acknowledged) ab, um das Ende seiner Datenübertragung anzuzeigen. Der Browser bestätigt dieses Paket mit ACK (16) und beendet seinen Kanal auf die gleiche Art (17). Nach der Bestätigung des Servers (18) ist die TCP-Session abgebaut und die Ressourcen stehen wieder zur Verfügung.

Nun kann der Browser den empfangenen html-Text untersuchen. Würde er Referenzen auf Bilder ("<img src=...>") enthalten, müßte der Browser jetzt noch alle Bilder anfordern. Dazu würde er parallel mehrere TCP-Sessions nach demselben Muster wie für die html-Seite öffnen und die Bilder abrufen. (Das ist die

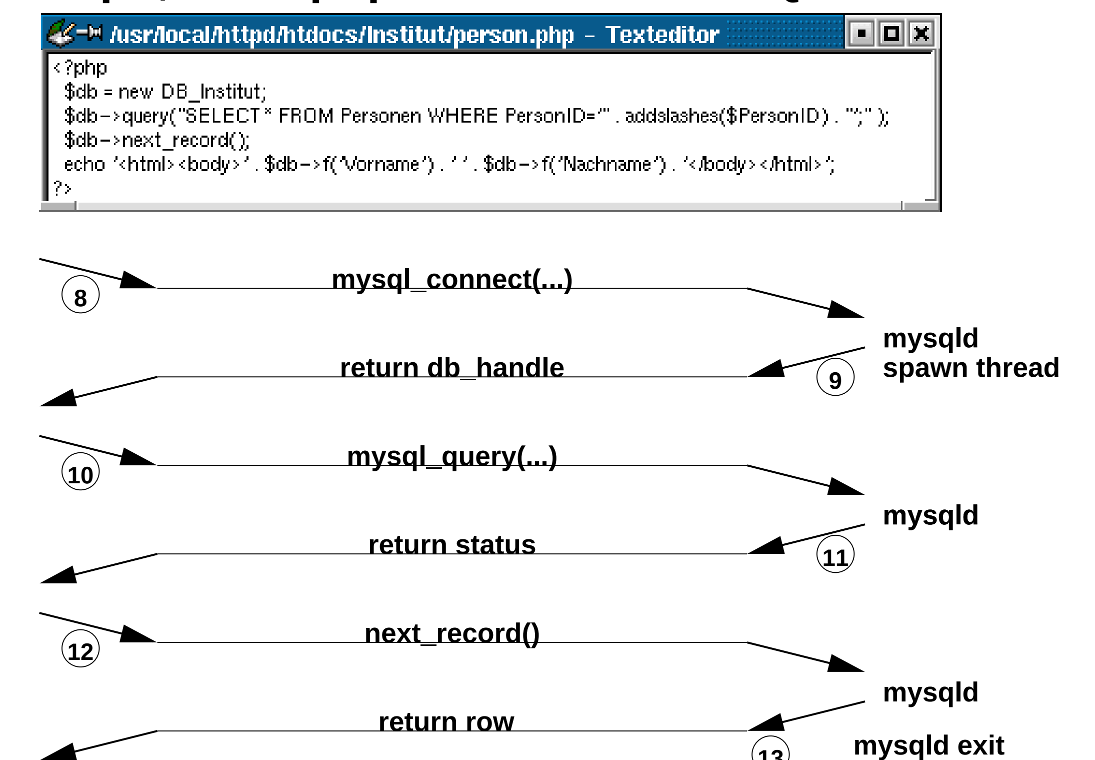


Abbildung 2.3: Zweiter Teil: Datenbankzugriff

am Beginn erwähnte Aneinanderreihung von einfachen Teil-Vorgängen, die durch wiederholte Anwendung komplexes Verhalten erreichen).

Eine typische Seite der Institutsdatenbank besteht aus bis zu 20 Bildern und benötigt zwischen 30 und 50 Datenbankzugriffen, um eine Seite aufzubauen (der Ablauf für solch einen Seitenabruf ist in keiner Weise *komplexer* als der hier beschriebene, er ist nur durch die große Anzahl an Schritten *komplizierter*, also schwer zu überblicken). In diesem einfachen Beispiel zeigt der Browser nun den gewünschten Namen an und wartet auf neue Eingaben des Benutzers; die Transaktion ist abgeschlossen.

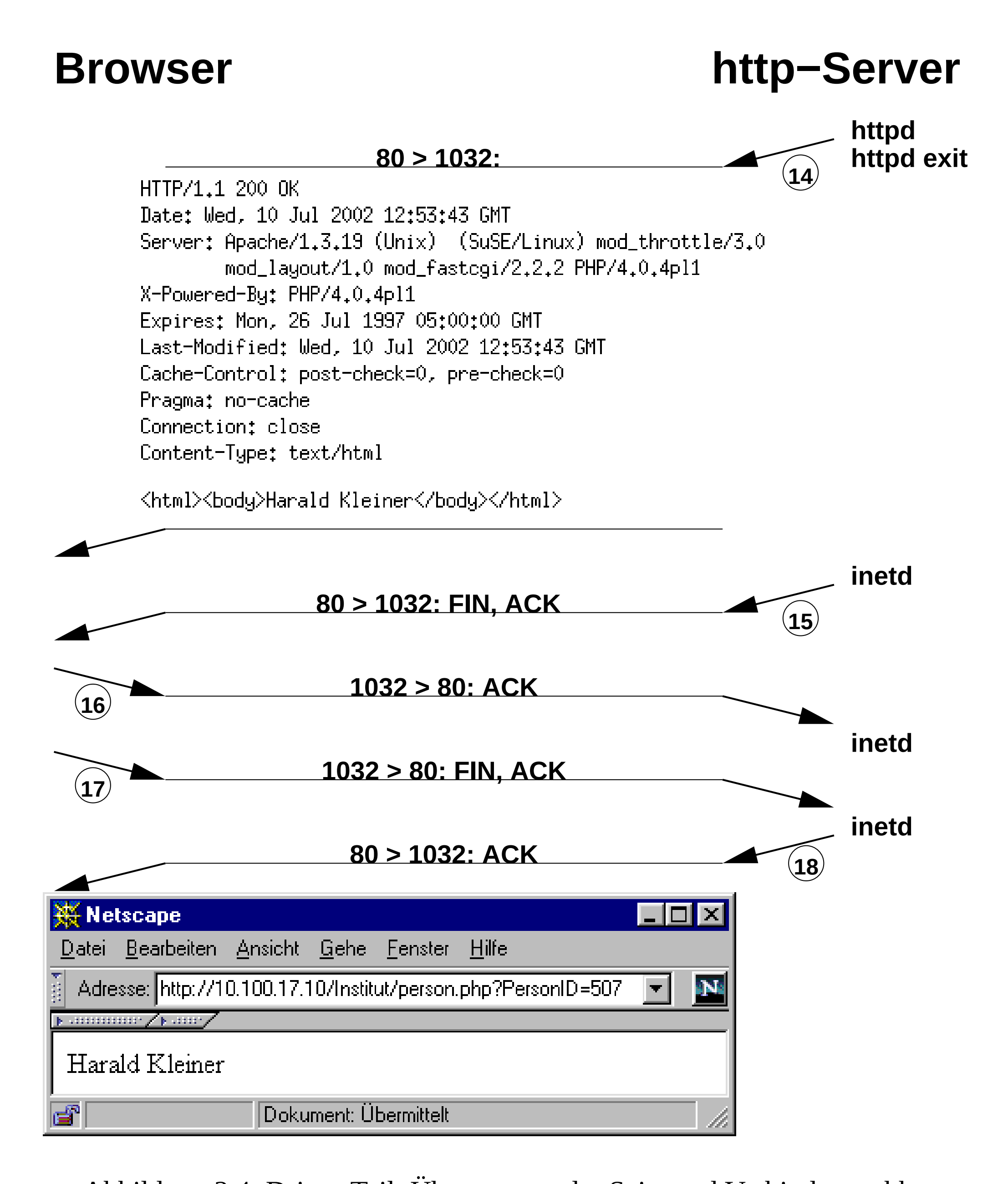


Abbildung 2.4: Dritter Teil: Übertragung der Seite und Verbindungsabbau

Kapitel 3

Entwicklungsprozeß

Dieses Kapitel behandelt den eingeschlagenen Weg, der zur Entwicklung der Web-Applikation geführt hat. Vom Datenbankentwurf über den Aufbau von Klassenbibliotheken bis hin zur Installation sollen jeweils die getroffenen Entscheidungen untermauert und begründet werden. Teile dieses Kapitels können als technische Dokumentation dazu dienen, den Leser an die verwendete Programmiermethodik heranzuführen, so daß er nach Erlangen eines Überblicks über den Quellcode in der Lage ist, selbst Erweiterungen vorzunehmen.

3.1 Datenbankentwurf

"Im Datenbankentwurf wird das Fundament gelegt, auf dem anschließend das Gebäude der Applikation errichtet werden soll."

Diese Weisheit sollte man nicht vergessen, wenn man in der Anfangsphase eines Projektes oft mühsame Diskussionen über die Organisation der Daten mit dem Auftraggeber zu führen hat. Es ist unbedingt notwendig, viel Zeit in einen ausgereiften Datenbankentwurf zu investieren, da ein schlechter Entwurf später kaum mehr korrigiert werden kann und so das gesamte Softwareprojekt belastet.

Es kommt beim Entwurf darauf an, die Entwurfsregeln zur Einhaltung der Normalformen *im Prinzip* einzuhalten und außerdem ein konsistentes Schema zur Benennung von Datenbankobjekten zu verwenden. Die Regeln zum Erreichen von Normalformen 'im Prinzip' einzuhalten bedeutet, sie in bestimmten Fällen zu verletzen, um einen einfacheren Tabellenentwurf oder spezielle Funktionalität zu erreichen; darauf wird im folgenden Abschnitt näher eingegangen.

Die konsistente Benennung von Datenbankobjekten macht sich erst später bezahlt, wenn man innerhalb der Abfragen dieselben Namen zu verwenden hat, die man selbst Wochen zuvor vergeben hat; dann ist die Verwendung kurzer und einfach zu merkender Namen sehr vorteilhaft. Hierfür existieren keine Normen, lediglich subjektive Präferenzen, die ebenfalls erläutert werden sollen.

3.1.1 Entwurfsregeln

Die Theorie des Datenbankentwurfs ([12], [13] und [14]) definiert Entities und Relations, also beliebige Objekte, die untereinander in Beziehung stehen. Sowohl Entities als auch (wenn auch selten) Relations können Attribute besitzen. Attribute sind einfache Werte, wie zum Beispiel ein Vorname, eine Länge oder ein Geburtsdatum.

Entities, Relations und die dazugehörigen Attribute lassen sich sehr anschaulich im Entity-Relationship-Diagramm (ER-Diagramm) darstellen. Das ER-Diagramm ist als Gesprächsbasis mit dem Auftraggeber ein wichtiges Hilfsmittel, da mit seiner Hilfe die Basis für das Datenbankmodell erarbeitet werden kann. Aus dem ER-Diagramm werden später die zu verwendenden Tabellen und die Abhängigkeiten der Tabellen untereinander abgeleitet.

In manchen Fällen ist nicht ganz offensichtlich, ob es sich bei einem Objekt um ein Attribut oder um eine eigene Entity handelt. Beispielsweise haben Personen ein Sitzzimmer, in dem sie ihren Arbeitsplatz haben. Man könnte nun einfach die Raumnummer als Attribut der Person zuordnen oder aber die Räume zu einer eigenständigen Entity erklären und eine Beziehung zwischen den Entities 'Räume' und 'Personen' herstellen (Abb. 3.1).

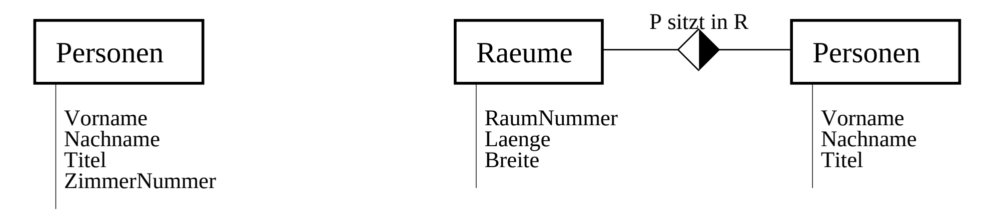


Abbildung 3.1: Realisierung als Attribut oder als Entity

Die erstere Methode sieht einfacher aus; sie kann auch durchaus ausreichend sein. Sie funktioniert jedoch nur dann, wenn keine weiteren Daten zum Sitzzimmer (wie beispielsweise Länge, Breite oder die Etage) gespeichert werden sollen.

Diese Daten sind *funktional* vom Raum *abhängig*, das bedeutet: Ist die Raumnummer gegeben, so folgen daraus direkt und eindeutig die Länge und die Breite des Raumes. Würde man nun zusätzlich zur Raumnummer auch Länge und Breite als Attribute der Person zuordnen, so wäre es denkbar, daß zwei Personen denselben Raum zugeordnet haben, jedoch unterschiedliche Größenangaben vermerkt sind. Dadurch würden sich die Ausmaße nicht mehr eindeutig ermitteln lassen.

Man erzeugt also in diesem Fall eine eigene Entity 'Räume', die alle Attribute der Räume enthält und die mit den Personen in Beziehung steht ('P sitzt in R'). Dadurch werden die Attribute Länge und Breite auch 'direkt beim Raum' abgespeichert. Diese beiden Attribute wären in der Personentabelle auch fehl am Platz, weil sie thematisch einfach nicht zur Person gehören.

Auch in einem zweiten Fall muß ein Attribut in eine eigene Entity umgewandelt werden. Das ist genau dann der Fall, wenn das Attribut mehrwertig sein müßte. Möchte man beispielsweise zu den Personen Telefonnummern speichern, so kann man die Telefonnummer als Attribut zur Person geben. Dadurch entsteht jedoch die Beschränkung, daß pro Person *genau eine* Telefonnummer gespeichert werden kann. Sollen mehrere Telefonnummern gespeichert werden können, so muß eine eigene Entity 'Telefonnummern' erstellt werden, die mit den Personen in Beziehung steht (siehe Tabelle 3.1). Auf diese Weise kann eine Person *beliebig viele* (also gar keine, eine, zwei ...) Telefonnummern haben. Es darf keinesfalls zugelassen werden, daß mehrere Telefonnummern in *ein* Feld eingetragen werden. Dadurch entstehen katastrophale Nebeneffekte: Die Felder können nicht mehr sinnvoll für eine listenartige Ausgabe verwendet werden und die eingegebenen Daten können nicht wieder voneinander getrennt werden[1]. Auch das Vorsehen von mehreren Attributen (Telefon1, Telefon2, Telefon3...) ist keine Lösung, da einerseits Speicherplatz verschwendet wird und andererseits trotzdem eine Person vier Telefonnummern besitzen kann.

Ein weiterer Aspekt zur Festlegung der Attribute ist Atomizität. Dabei ist zu beachten, daß das Zusammenfügen von Daten (durch Verkettung von Strings) besonders einfach zu bewerkstelligen, das Aufspalten von Daten in Unterabschnitte in der Praxis jedoch kaum möglich ist. Als Beispiel sei hier der Name von Personen aufgeführt. Würde man nur ein Feld für den gesamten Namen samt Titel vorsehen, so wäre die Schreibweise (die Reihenfolge von Anrede, Titel, Vor- und

[1] Die Verwendung speziell vereinbarter Trennzeichen für solche Zwecke ist keine Lösung, weil sie ein inakzeptabel hohes Maß an Disziplin bei der Dateneingabe verlangt.

Personen

Nr.	Nachname	Telefon
1	Huber	123, 456
2	Müller	876

→

Personen

Nr.	Nachname
1	Huber
2	Müller

Telefonnummern

PersonNr.	Telefon
1	123
1	456
2	876

Tabelle 3.1: Ausgliederung der Telefonnummern

Nachname) nicht vorgegeben und würde sich in den einzelnen Datensätzen – je nach Geschmack der eingebenden Person – unterscheiden.

Es wäre dann nicht mehr möglich, Namenslisten nach Vor- oder Nachname zu sortieren oder beispielsweise eine persönliche Anrede (‘Sehr geehrte. . . ’ oder ‘Sehr geehrter. . . ’) etwa für einen Brief automatisch zu erstellen. Eine nachträgliche Trennung ist auf automatischem Wege kaum möglich; man denke nur an Doppelnamen oder Vornamen wie ‘Marie Sophie’. Daher müssen Attribute schon beim Entwurf so weit in Teil-Attribute aufgespaltet werden, bis atomare Einheiten erreicht sind. Das führt dazu, daß in der Institutsdatenbank sechs Felder zur vollständigen Speicherung des Namens verwendet werden (Anrede, Vorname, Nachname, Titel in Langform, Titel in Kurzform sowie Amtstitel), die jedoch zur Ausgabe beliebig kombiniert werden können. Bei der Aufspaltung ist eine genaue Analyse der Daten notwendig, damit kein nicht-atomares Attribut übersehen wird.

Die Theorie definiert zur Kategorisierung von Datenbankentwürfen sogenannte *Normalformen*. Mit großem formalen Aufwand werden dort Begriffe wie Schlüssel, Abhängigkeit, transitiv oder Primattribut definiert. Hier wird nun versucht, ohne Verwendung dieser Begriffe für die Praxis geeignetere und leichter verständliche Regeln zu formulieren. Außerdem wird auch gleich auf die Fälle hingewiesen, in denen bewußt von den theoretischen Vorschriften abgewichen wird.

Erste Normalform: Läßt sich ein Attribut in weitere Teile aufspalten, so müssen *grundsätzlich* die Einzelteile voneinander getrennt als Einzelattribute gespeichert werden. Zu dieser Regel gibt es nur wenige Ausnahmen: beispielsweise Adressen. Eine Adresse besteht zwar aus Straßenname und Hausnummer, wenn es jedoch mit absoluter Sicherheit nicht nötig ist, auf die Straße bzw. die Hausnummer alleine zuzugreifen, kann ein Feld ‘Adresse’ definiert werden, das Straße

und Hausnummer gemeinsam enthält.

Zweite Normalform: Ein Attribut wird zu einer Entity, wenn das Attribut mehrere Werte aufnehmen soll (beispielsweise Telefonnummern). Das Attribut wird aus seiner ursprünglichen Entity herausgenommen und statt dessen als neue, eigene Entity mit der Ursprünglichen in Beziehung gesetzt. Jede eingetragene Telefonnummer wird so zu einer eigenen Zeile in der 'Telefonnummern'-Tabelle.

Dritte Normalform: Ein Attribut (z.B. Raumnummer) wird zu einer Entity, wenn noch weitere von diesem Attribut abhängige Attribute existieren (z.B. die Länge des Raumes). Das Attribut samt seiner abhängigen Attribute wird aus der ursprünglichen Entity (z.B. Personen) herausgenommen und mit ihm in Beziehung gesetzt. Diese Regel muß besonders streng befolgt werden, damit Inkonsistenzen vermieden werden. Trotzdem gibt es Ausnahmen. Bei der Anschrift einer Person ist der Ortsname funktional von der Postleitzahl abhängig. Beide Felder werden aber als Attribute der Person zugeordnet. Würde man eine eigene Entity 'Orte' anlegen, so bräuchte man nur noch die Postleitzahl eingeben und der Ortsname wäre damit automatisch festgelegt. Andererseits würde sich ein Tippfehler in der Postleitzahl besonders stark auswirken, da sofort ein völlig anderer Ortsname entstehen würde. Außerdem besteht bei manueller Eingabe die Möglichkeit, den Ortsteil anzugeben.

Der Vollständigkeit halber werden an dieser Stelle noch ein paar Worte zur Komplexität von Beziehungen angefügt. Darunter versteht man die mögliche Anzahl von Objekten, die auf jeder Seite einer Beziehung teilnehmen können. Abbildung 3.2 zeigt die drei Möglichkeiten, die für zweiwertige Beziehungen (also zwei beteiligte Entities) existieren, sowie ein Beispiel für eine dreiwertige Beziehung.

Der erste Fall stellt eine 1:1-Beziehung dar. Das bedeutet, daß auf jeder Seite der Beziehung genau eine Instanz der Entities beteiligt sein kann. In diesem Fall also wird eine Abteilung von *genau einer* Person geleitet, und eine Person kann *genau eine* Abteilung leiten.

Bei der zweiten Beziehung handelt es sich um eine 1:n-Beziehung. Dabei wird jeweils eine Entity von der 1-Seite mit vielen Entities auf der n-Seite verbunden. Die n-Seite wird durch die schwarz ausgefüllte Hälfte markiert. Im Beispiel können in einem Raum *mehrere* Personen sitzen, jede Person sitzt aber in *genau einem* Raum.

Die dritte Beziehung stellt den Fall einer n:m-Beziehung dar. Dabei können von beiden beteiligten Entities beliebig viele an der Beziehung teilnehmen. Im Beispiel nehmen also *mehrere* Personen an einem Projekt teil und eine Person kann auch an *mehreren* Projekten teilnehmen.

Die letzte Beziehung schließlich gehört nicht zu den Grundtypen. Sie wird hier erläutert, weil sie im ER-Modell der Institutsdatenbank vorkommt. Diese Beziehung stellt dar, daß Personen *mehrere* Tätigkeiten ausführen können. Zu jeder Tätigkeit, die eine Person ausführt, gehört jedoch *genau ein* Betreuer (eine andere Person). Ein Betreuer kann *mehrere* Tätigkeiten betreuen, zu jeder Tätigkeit gehört jedoch *genau eine* betreute Person. Schließlich gehört zu jeder Tätigkeit *genau eine* ausführende Person und *genau ein* Betreuer[2].

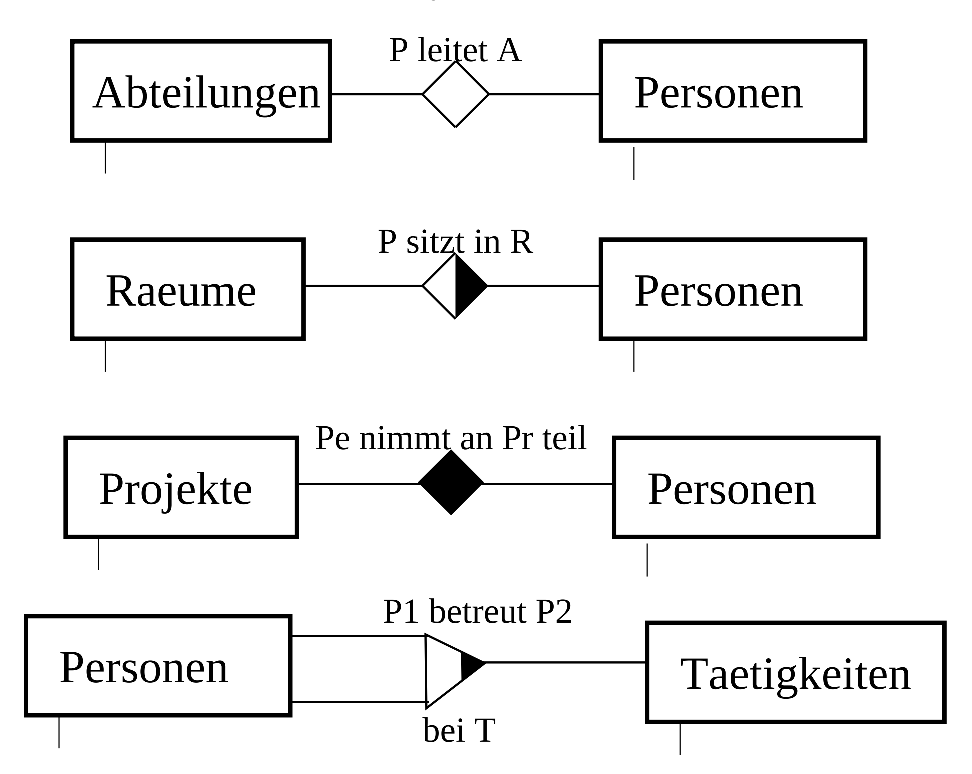


Abbildung 3.2: Grade der Komplexität von Beziehungen

Beziehungen werden durch Austausch von Schlüsseln (Attribut, das Zeilen in der jeweiligen Tabelle eindeutig definiert, häufig einfach nur eine Zeilennummer) realisiert. Bei 1:1-Beziehung kann in eine der beiden Entities das Schlüsselfeld

[2]Man kann jede Beziehung von jeder der beteiligten Entities aus betrachten; die angegebenen Komplexitäten müssen unabhängig vom Blickwinkel den realen Gegebenheiten entsprechen.

der anderen Entity aufgenommen werden. Dabei spielt es keine Rolle, welcher der beiden Schlüssel verwendet wird. Bei 1:n-Beziehungen wird bei der Entity auf der n-Seite das Schlüsselfeld der Entity der 1-Seite als Attribut hinzugefügt. Es wird also beispielsweise bei den Personen das Schlüsselfeld der Räume eingetragen, um das Sitzzimmer zu modellieren.

n:m-Beziehungen können auf diese Art nicht realisiert werden, da ja auf beiden Seiten jeweils mehrere Schlüsselwerte einzutragen wären. Statt dessen wird zum Aufbau von n:m-Relationen eine eigene Entity erstellt, die die beiden Schlüsselfelder enthält. Diese Entity kann auch weitere Attribute enthalten, die zur Beziehung gehören – etwa Beginn und Ende der Teilnahme am Projekt.

3.1.2 Konventionen

Im Folgenden sollen die Regeln dargelegt werden, die bei der Umsetzung des ER-Diagramms in Tabellen befolgt wurden. Es handelt sich dabei um Konventionen, die sich durch oftmaligen Gebrauch entwickelt und bewährt haben, die allerdings nicht als allgemein gültige Norm vorliegen.

Ziel dieser Konvention ist es, die Benennung der Datenbankobjekte konsistent und knapp auszuführen, so daß der Name eines bestimmten Attributes, eines Schlüsselfeldes oder einer Tabelle möglichst kurz und möglichst ohne Blick auf das ER-Diagramm leicht zu merken oder zu ‘erraten’ ist. Dies macht sich bei der Erstellung der Software bezahlt, da die Namen im Programmcode intensiv verwendet werden müssen.

Alle Bezeichnungen werden ausnahmslos im Standard-ASCII-Zeichensatz vergeben. Das bedeutet, daß Umlaute durch ihre Zweibuchstabenrepresentationen (ae, oe, ue sowie ss statt scharfem ß) ersetzt werden. Als einziges Sonderzeichen wird der Underscore (_) nur in besonderen Fällen sparsam verwendet, um die Lesbarkeit zu verbessern.

Grundsätzlich werden alle Bezeichner möglichst kurz gewählt, um Tipparbeit einzusparen und die Übersicht zu steigern – trotzdem darf die Verständlichkeit nicht leiden. Müssen Worte zusammengesetzt werden, so wird das zweite Wort mit Großbuchstaben begonnen, wenn es sich um eine unübliche Zusammensetzung handelt, zum Beispiel:

`‘AnschlussTyp’ ‘NutzungDekan’` aber `‘Geburtsdatum’ ‘Amtstitel’`

Da (zumindest unter Linux und innerhalb einer SQL-Anweisung) Groß- und Kleinschreibung beachtet wird, ist auch in diesem Zusammenhang auf Konsistenz

zu achten.

Tabellen: Tabellennamen werden immer im Plural vergeben, um anzuzeigen, daß die Tabelle eine Sammlung gleichartiger Objekte desselben Namens ist (die Tabelle ‘Raeume’ ist eine Sammlung von ‘Raum’-Objekten). Der Tabellenname muß weiters die darin gespeicherten Objekte eindeutig und klar kennzeichnen. Das bedeutet auch, daß alle in einer Tabelle gespeicherten Felder (die Spalten) in direktem Zusammenhang mit dem Tabellennamen stehen müssen: Es wäre beispielsweise falsch, in der Personentabelle die Etage des Sitzzimmers abzuspeichern. Diese Information ‘gehört nicht’ in die Personentabelle, da die Personen nicht direkt mit Etagen, sondern mit den Räumen in Beziehung stehen (siehe Abb. 2.1).

Schlüsselfelder

Grundsätzlich vertritt der Autor die Überzeugung, daß Attribute (also Daten des Objektes) nur in den seltensten Fällen als Schlüsselfelder geeignet sind. Dies läßt sich durch zwei Gründe untermauern: Zuerst einmal finden sich in der realen Welt kaum Attribute, die zwingend vorhanden und gleichzeitig *wirklich* eindeutig sind. Für Personen wäre beispielsweise der Name als Schlüssel völlig ungeeignet, selbst Sozialversicherungsnummern sind nicht in allen Fällen eindeutig oder überhaupt vorhanden. Und weiters besteht die Möglichkeit, daß Attribute ihren Wert im Laufe der Zeit ändern. Dann müßte in der gesamten Datenbank jedes Vorkommen des Schlüssels ausgebessert werden, um die Konsistenz zu erhalten.

Aus diesen Gründen werden Schlüssel in der Institutsdatenbank prinzipiell als Zeilennummernfeld (‘auto-increment’) ausgeführt. Dadurch trägt der Schlüssel keine Information – er dient nur zur Identifikation des Datensatzes und gelangt niemals sichtbar an die Benutzeroberfläche. Der dadurch zusätzlich verbrauchte Speicherplatz kann als vernachlässigbar bezeichnet werden; außerdem ist die Suche nach Zahlen in der Datenbank deutlich effizienter als die Suche nach Zeichenketten.

Benennung: Schlüsselfeldnamen sollen aus dem Tabellennamen abgeleitet werden können. Deshalb ist der Schlüsselname derselbe wie der Tabellenname, jedoch im Singular und mit nachgestelltem ‘ID’ (Identification, sprich ‘Ei-Di’).

In manchen Fällen wird jedoch noch ein Zeichen eingefügt, um die Aussprache zu vereinfachen: beispielsweise

`'FaxKlappenID'` statt `'FaxKlappeID'`

Fremdschlüsselfelder (Schlüsselfelder, die eine Beziehung zu einer anderen Tabelle realisieren, z.B. die RaumID der Personen) werden manchmal umbenannt, um die Verständlichkeit wiederherzustellen. Zu Rechnern wird beispielsweise die verantwortliche Person gespeichert. Das Fremdschlüsselfeld heißt jedoch nicht `Rechner.PersonID`, sondern `Rechner.VerantwortlicherID`, weil ein Rechner 'keine Person besitzt', sondern 'einen Verantwortlichen hat'.

Diese Methode muß aber *sparsam* verwendet werden, da gleiche Dinge auch gleich heißen sollen.

Attribute sollen möglichst über die gesamte Datenbank hinweg eindeutig benannt sein. Es wurde vermieden, allgemeine Namen wie 'Text', 'Name' oder 'Beschreibung' zu verwenden, da nicht eindeutige Attribute bei Verwendung mit dem Tabellennamen qualifiziert werden müssen. Deshalb heißt es auch beispielsweise `AnschlussTypen.AnschlussTyp` und nicht `AnschlussTypen.Name`, obwohl das eigentlich ein Verstoß gegen das Datenbankmodell ist: Ein AnschlussTyp hat nämlich eigentlich keinen AnschlussTyp als Eigenschaft, sondern einen Namen. So gesehen ist daher die Bezeichnung falsch gewählt. Um aber die Qualifizierung mit dem Tabellennamen zu vermeiden, wird dieser praxisnahe Weg gewählt.

3.2 Programmentwicklung

Dieser Teil beschreibt den Kern des Projektes, nämlich die Entwicklung der Software. Im Kapitel 2.1 wurden die Anforderungen aufgelistet, die an die Benutzeroberfläche gestellt werden. Die enthaltenen Anforderungen, wie zum Beispiel die Überprüfung der Zugriffsrechte oder die Plausibilitätsprüfungen für eingegebene Daten würden ohne Modularisierung und Verkapselung von Basisfunktionen zu einem unüberschaubaren und kaum wartbaren Programmcode führen. Deshalb wurde von Beginn an auf objektorientiertes Programmieren und Klassenbildung geachtet und häufig verwendete Programmteile nach und nach in die Hilfsklassen aufgenommen. Dadurch konnte die Programmlänge auf durchschnittlich 150 Zeilen pro Formular reduziert werden.

Zunächst werden die verwendeten Programme und Bibliotheken vorgestellt, auf denen die selbsterstellten Klassen aufbauen. Danach folgt der gesamte Ent-

wicklungsprozeß bis zur Installation und Wartung.

3.2.1 Werkzeuge

Quanta: Die Wahl der richtigen Werkzeuge beginnt beim Editor. Für die Institutsdatenbank wurde Quanta verwendet, ein Editor für KDE. Selbstverständlich existieren viele weitere sehr gute Editoren – die folgenden Eigenschaften von Quanta haben sich jedoch sehr bewährt: Quanta bietet eine integrierte Hilfe für html, php und JavaScript und kann mehrere Dateien parallel geöffnet halten. Der größte Vorteil ist jedoch die Syntaxhervorhebung: Da auch Zeichenketten andersfarbig darstellt werden, ist es besonders einfach, fehlende Anführungszeichen oder ähnliche Tippfehler auf einen Blick zu erkennen.

Objektorientierung - php: Die benötigte Modularisierung läßt sich besonders komfortabel und anschaulich durch Objektorientierung erreichen. Glücklicherweise unterstützt und fördert php seit Version 4 das Objektorientierte Programmieren.

phplib: Teile der benötigten Funktionalität beruhen nicht auf völlig neuen Anforderungen, die noch kein Projekt zuvor zu erfüllen hatte. PHPLib ist eine Sammlung von Klassen, die Basisfunktionalität für verschiedene Bereiche der Webprogrammierung bereitstellen. Besonders hervorzuheben ist hier die Form-Klasse. Sie beinhaltet die Erstellung von form-Tags in html und übernimmt auch die Plausibilitätsprüfung der eingegebenen Daten. Erst die Erweiterung dieser Klasse entsprechend den Bedürfnissen hat die Programmierung auf dem benötigten hohen Niveau ermöglicht.

gd ist eine Bibliothek, mit der man Bilder zur Laufzeit erstellen und bearbeiten kann. Die GD-Bibliothek läßt sich besonders effizient zum Erstellen von graphischen Schaltflächen verwenden: Da die Schaltfläche bei jedem Aufruf automatisch neu erstellt wird, ist der Aufwand zur Änderung der Beschriftung minimal.

LaTeX: Diese Ausarbeitung wurde selbstverständlich in LaTeX verfaßt. Das garantiert höchste Stabilität und Satzqualität auch bei langen Texten mit Bildern und Querverweisen, wie man sie von keinem Textverarbeitungssystemen bekommt.

Die Abbildungen wurden mit XFig erstellt. XFig ermöglicht die Erstellung von Linienzeichnungen mit Text und eingebetteten Bildern. Durch die verschachtelte Blockbildung können Zeichnungsteile gruppiert und gemeinsam verschoben werden. Der Export erfolgt mittels Encapsulated PostScript (eps) vektororientiert, dadurch entstehen keine Qualitätsverluste bei der Größenanpassung.

3.2.2 Aufbau und Zusammenspiel der Klassen

Als Grundlage der Webapplikation dient eine Auswahl von Klassen aus der PHPLIB. PHPLIB selbst ist als frei verfügbare Sammlung von php-Klassen unter phplib.sourceforge.net erhältlich. Die Bibliothek ist sehr ausgereift und dennoch überschaubar, was eigene Anpassungen unterstützt. Zu PHPLIB existiert auch eine sehr gute Dokumentation [15], die vor Anwendung der Klassen unbedingt durchgearbeitet werden sollte.

Zunächst soll der Aufbau der Klassenhierarchie sowie der entstandene Mehrwert durch die Erweiterungen dargestellt werden; danach folgt eine Darstellung über den Programmablauf beim Aufruf einer Formularseite.

Klassenhierarchie: Abbildung 3.3 zeigt die Beziehungen der Klassen untereinander. Die grau unterlegten Klassen stammen aus PHPLIB und wurden unverändert übernommen. Alle benötigten Anpassungen sind in abgeleiteten Klassen realisiert (Ableitungen werden durch kräftige Pfeile dargestellt; die dünnen Pfeile zeigen die Anwendung der Klassen an). Dadurch werden die PHPLIB-Dateien nicht verändert (die Klassenbibliothek kann von mehreren Web-Applikationen gleichzeitig verwendet werden).

Im Zentrum der Applikation stehen die Dokumentklassen (`*_doc`), die alle von der gemeinsamen Mutterklasse `Site` abgeleitet sind. Jedes einzelne Dokument der Applikation ist in einer eigenen gleichnamigen Datei abgelegt. Da die einzelnen Seiten von einer gemeinsamen Klasse abstammen, ist ihnen allen derselbe systematische Aufbau und somit derselbe Programmablauf gemein.

PHPLIB-Funktionalität: Zur Funktion der PHPLIB-Klassen sei der Leser auf die Original-Dokumentation [15] verwiesen; das wichtigste soll jedoch hier kurz zusammengefaßt werden.

Datenbankzugriff: `DB_Sql` und die davon abgeleitete Klasse `DB_Institut` ermöglichen den Zugriff auf eine SQL-Datenbank. Zu PHPLIB gehören mehrere

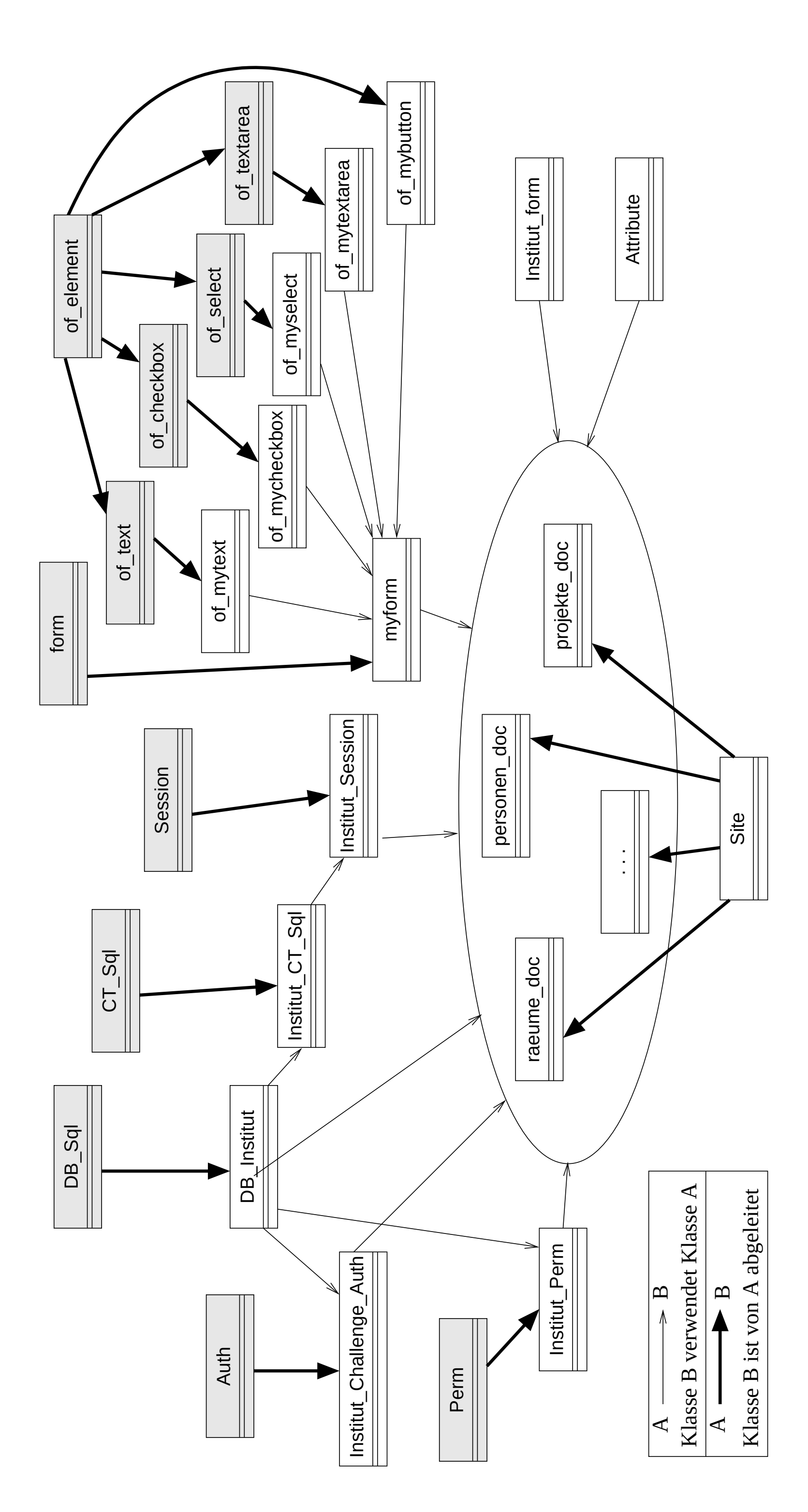


Abbildung 3.3: Hierarchie der Klassen

Dateien, die die Klasse `DB_Sql` für verschiedene Datenbankserver implementieren (`db_mysql.inc`, `db_odbc.inc` und andere) - je nach vorhandenem Server ist eine davon auszuwählen. Die Klasse erledigt den Verbindungsauf- und Abbau zum Datenbankserver und wird hauptsächlich über die Methoden `query()` und `next_record()` angesprochen. Die abgeleitete Version `DB_Institut` beinhaltet neben einigen zusätzlichen Methoden noch die Anmeldedaten für den Datenbankserver. `DB_Institut` wird sehr häufig benötigt, um Abfragen an den SQL-Server zu senden.

Session, Authentifizierung und Privilegien: `CT_Sql` und die abgeleitete `Institut_CT_Sql` bilden – aufbauend auf die Datenbankklasse `DB_Institut` eine Vermittlerklasse zu `Institut_Session`. Um die `Session`-Klasse unabhängig von der darunterliegenden Speichermethode für die Daten zu gestalten, gibt es in PHPLIB mehrere solche 'Container'-Klassen (für SQL, LDAP, shared memory und DBM). Um MySQL zum Speichern der Session-Daten zu verwenden, wird in `Institut_CT_Sql` der Name der Datenbankklasse angegeben und die von `Session` abgeleitete Klasse `Institut_Session` wird so konfiguriert, daß sie `Institut_CT_Sql` als Container verwendet.

Die `Session`-Klasse bildet ihrerseits nun die Basis für die Benutzerauthentifizierung (`Auth`) und das Permission-Management (`Perm`): Die `Session`-Klasse erzeugt SessionIDs, also eindeutige Zeichenketten, anhand derer ein neuerlicher Seitenabruf als Teil einer bestehenden Session wiedererkannt werden kann. Dadurch sind alle Seitenabrufe eines Benutzers eindeutig zuzuordnen und von den Abrufen der anderen Benutzer trennbar. Außerdem bietet `Session` die Möglichkeit, Programmvariablen zu 'registrieren'. Die Werte der registrierten Variablen werden am Ende des Programmablaufes automatisch durch die `Session`-Klasse in die Datenbank geschrieben und beim nächsten Seitenabruf aus der Datenbank wiederhergestellt. Für den Programmierer erscheinen registrierte Variablen also wie statische globale Variablen: Die Parallelität und die zeitliche Verschachtelung der Seitenabrufe ist somit aufgehoben.

Mit Hilfe der `Session`-Klasse werden die Seitenabrufe jedes Benutzers durch eine gemeinsame SessionID zusammengefaßt. Mit Hilfe von `Auth` kann nun festgestellt werden, *wer* der Benutzer eigentlich ist. `Auth` stellt also eine Beziehung zwischen einer aktuellen Sitzung (einer `SessionID`) und einem Benutzerdatensatz aus der `user`-Tabelle (einer `userID`) her. Dazu wird ein Login-Formular bereitgestellt, in dem der Benutzer seinen Namen und sein Paßwort angeben kann, um sich zu authentifizieren. Erst wenn der angegebene Name und das Paßwort

mit den in der Datenbank abgelegten Benutzerdaten übereinstimmen, gilt der Benutzer als authentifiziert und die `userID` steht zur Ermittlung der Berechtigungen zur Verfügung. Die abgeleitete Klasse `Institut_Challenge_Auth` implementiert übrigens ein Challenge-Response Verfahren zur Authentifizierung. Dabei wird die MD5-Prüfsumme von Benutzername, Paßwort und einer zufälligen Zeichenkette (der Challenge) berechnet, und als Antwort zum Server retourniert. Der Server ermittelt dieselbe Prüfsumme mit dem Paßwort aus der `user`-Tabelle und überprüft, ob die beiden Prüfsummen übereinstimmen. Dadurch wird das Paßwort nie (im Klartext) übertragen und kann nicht abgehört werden.

Die 'Krönung' dieser Hierarchie bildet nun `Perm`: Diese Klasse ordnet jedem authentifizierten Benutzer (jeder `userID`) eine oder mehrere Privilegienstufen zu. Weiters kann mittels Methoden der `Perm`-Klasse abgefragt werden, ob der Benutzer ein spezielles Set von Privilegien besitzt, das zum Betrachten der aktuellen Seite gerade benötigt wird. Die abgeleiteten Klassen `Institut_Challenge_Auth` und `Institut_Perm` implementieren noch weitergehende Funktionalität, da die Privilegien der Benutzer für jedes Datenbankfeld extra als das Maximum der Privilegien aus allen Gruppenzugehörigkeiten ermittelt werden müssen.

Formulardesign: Mit Hilfe der `form`-Klasse läßt sich die Erstellung von html-Formularen samt anschließender Validierung (Plausibilitätsprüfung) stark vereinfachen. Zunächst wird festgelegt, welche Steuerelemente (Textfeld, Auswahlliste o.ä.) verwendet werden sollen. Davon unabhängig wird in einem zweiten Schritt das Layout des Formulars bestimmt. Die Erstellung des html-Quelltextes für die Steuerelemente übernimmt die `form`-Klasse. Dadurch kann besonders viel Quelltext eingespart werden und die Übersichtlichkeit wird stark verbessert.

Die `form`-Klasse ist selbst modular aufgebaut und besteht aus Basisklassen zur Implementierung der einzelnen Steuerelemente (Abb. 3.3). Um das Verhalten an die Bedürfnisse anzupassen und die Abstraktion noch weiter voranzutreiben, werden jeweils abgeleitete Klassen verwendet: In der Klasse `myform` wird die Methode `add_text_element()` definiert, die das Erstellen von Textfeldern stark vereinfacht, indem sie anhand des Datentyps den Validierungsausdruck für das Feld automatisch festlegt. Die Methode `add_optionlist()` fügt dem Formular eine Auswahlliste hinzu. Dabei wird die Methode `create_optionlist()` zu Hilfe genommen, um die angegebene SELECT-Abfrage auszuführen und das Array von Optionswerten für die `add_element()`-Methode zu erstellen. `add_optionlist()` ist eine sehr kompakte Methode, und erfüllt eine häufig verwendete Funktion. `show_element_tab()` dient schließlich der Ausgabe der Steuerelemente in tabel-

larisch angeordneten Formularen. Sie überprüft die nötigen Berechtigungen und gibt das Steuerelement mit der zugehörigen Beschriftung aus.

Eigene Klassen: Zur Automatisierung wiederkehrender Aufgaben wurde die Klasse `Institut_form` erstellt. Sie kapselt verallgemeinerte Programmsegmente, die aus den Formulardateien ausgelagert wurden. `Institut_form` enthält Methoden zur Erstellung von Auswahllisten und Befehlsschaltflächen, sowie zum Ausführen der Abfragen.

Mit Hilfe der `Attribute`-Klasse werden den Formularen anwählbare Felder (Checkboxen) hinzugefügt, deren Beschriftung für jedes Formular frei gewählt werden kann. Diese Funktion verbessert die Erweiterbarkeit erheblich, weil pro Tabelle bis zu 31 verschiedene Ja/Nein-Werte gespeichert werden können, deren Bedeutung erst während der Anwendung der Applikation festgelegt wird. Durch die Verkapselung der benötigten Funktionen in die `Attribute`-Klasse beschränkt sich der Mehraufwand auf etwa fünf zusätzliche Zeilen Quellcode pro Formular.

Programmablauf: Um die Interaktion der verschiedenen Klassen und die involvierten Dateien deutlich zu machen, wird hier kurz die Reihenfolge der Programmausführung dargestellt. Dieser Ablauf ersetzt den Mittelteil des Fallbeispiels aus Kapitel 2.3.2.

Wird eine Datei vom Server abgerufen (in diesem Beispiel `personen.php`), so führt das php-Modul zuerst automatisch die Datei `prepend.php3` aus. Dies wird durch die in `php.ini` angegebene Konfigurationsvariable `auto_prepend_file` konfiguriert[3]. `prepend.php3` bindet die benötigten PHPLIB-Dateien ein und definiert Konstanten (siehe Tabelle 3.2). Anschließend beginnt die Ausführung der eigentlich angeforderten Datei. Diese enthält als erstes einen Verweis auf `pagestart.inc`. Dort werden die selbstdefinierten Klassen für Session, Authentifizierung und Permissions instantiiert und gestartet. Zu diesem Zeitpunkt stellt nun die `Session`-Klasse die registrierten Variablen wieder her.

Anschließend kehrt die Ausführung wieder zurück zu `personen.php`. Hier wird nun die Dokumentklasse von `Site` abgeleitet und die Methode `inhalt()` zur Erstellung des Seiteninhalts implementiert. Dabei entstehen durch die Anwendung der jeweiligen Klassen Sprünge in die Dateien `local.inc` sowie

[3]Sollte die Verwendung von `auto_prepend_file` nicht möglich sein, so kann `prepend.php3` auch durch die Datei `pagestart.inc` eingebunden werden, da diese in jeder Datei verwendet wird.

Datei	Programmsegment, Anmerkung
prepend.php3	inkludiert db_mysql.inc, ct_sql.inc, session.inc, auth.inc, perm.inc, oohforms.inc, page.inc und Institut.inc.
personen.php	include(pagestart.inc)
→ pagestart.inc	page_open() startet die von Session, Auth und Perm abgeleiteten Klassen. Die Session-Klasse stellt registrierte Variablen wieder her.

personen.php←	class personen_doc extends Site{
	Diverse Sprünge in local.inc und Institut.inc
	}
	$doc = new personen_doc;
	$doc->p();
	page_close();

→ page.inc	startet $sess->freeze()

Tabelle 3.2: Aufrufreihenfolge

Institut.inc: local.inc enthält alle von PHPLIB abgeleiteten Klassen, die selbstdefinierten Klassen sind in Institut.inc zu finden.

Die Programmausführung springt allerdings zuerst an das Ende der Datei (Klassendefinitionen sind wie Typdeklarationen). Dort wird die eben definierte Dokumentklasse instantiiert und die Methode p() ausgeführt, um den Seiteninhalt auszugeben. Am Ende folgt der Aufruf von page_close(). Dadurch werden die aktuell gültigen Werte der registrierten Variablen in der Datenbank gespeichert ($sess->freeze()).

3.2.3 Aufbau der Formulare

Site-Klasse

Wie in der Aufgabenstellung gefordert, sollen alle Formulare der Applikation dasselbe Layout und gemeinsame Navigationselemente besitzen. Die Navigationselemente (Menüs und Links zu anderen Seiten) dürfen jedoch nicht starr sein, sondern müssen an jede Seite angepaßt sein – abhängig von der Position der Seite in der hierarchischen Organisation. Als scheinbar einfache Lösung hätte man die Navigationselemente in jede einzelne Datei als fixen Bestandteil integrieren

können. Dabei müßte jedoch die Konsistenz während der Erweiterung der Applikation ständig durch Anpassungen in allen Dateien wiederhergestellt werden. Um diesen Aufwand zu vermeiden, wurde ein vollautomatisches System entwickelt, um die Navigationselemente zu erzeugen; dabei ist die `Site`-Klasse entstanden.

Die `Site`-Klasse beinhaltet jene Bestandteile, die auf allen Formularen zu finden sind. Sie gibt die Seitenaufteilung vor und erstellt die Navigationselemente automatisch. Da alle Formulare der Applikation durch Klassen erstellt werden, die von `Site` abgeleitet sind und nur jene Methoden überschrieben werden, die den eigentlichen Seiteninhalt sowie die Seitenüberschrift ausgeben, ist die Konsistenz der Seiten automatisch und 'wartungsfrei' sichergestellt.

In Abbildung 3.4 ist das vorgegebene Seitenlayout mit den Methoden zur Erstellung der jeweiligen Inhalte dargestellt. Links wird eine Baumansicht der gesamten Applikation erzeugt, im rechten Bereich oben und unten eine Liste mit Tabreitern zur Navigation auf der aktuellen Ebene. Der schraffierte mittlere Bereich steht für den eigentlichen Inhalt und die Seitenüberschrift zur Verfügung (der Rahmen wird durch eine html-Tabelle realisiert, die ohne Angabe fixer Längen vom Browser passend vergrößert werden kann). Am Ende der Seite gibt `footer()` Statusinformationen aus, die sich zur Fehlersuche verwenden lassen (gesteuert durch die Konstanten `be_verbose` sowie `be_more_verbose`, definiert in `prepend.php3`). Die Methode `p()` erzeugt das Gerüst der Seiten und ruft alle weiteren Methoden auf, damit diese 'ihre' Inhalte ausgeben können.

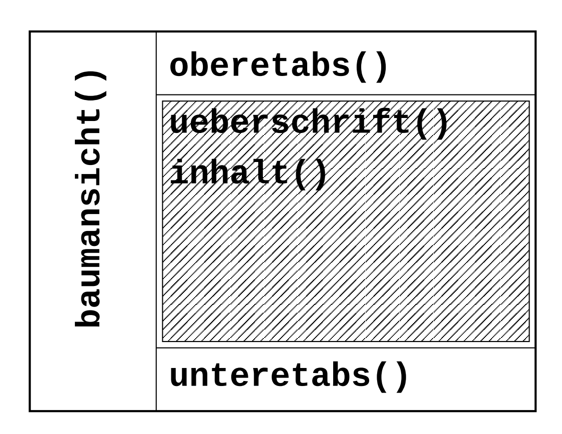


Abbildung 3.4: Das vorgegebene Layout der Formulare

Die Methoden zur Erstellung der Navigationselemente verwenden eine Repräsentation der Seitenhierarchie, die neben der Struktur der Applikation auch die Dateinamen, die Beschriftung der Links sowie die benötigten Privilegien beinhaltet. Diese Informationen sind in der `tree`-Datenstruktur (`Institut.inc`) zusammengefaßt.

`tree` ist ein zweidimensionales assoziatives Array, das entspricht einem Array von struct-Datenstrukturen nach der Notation von C – an Stelle einer Deklaration erfolgt die Benennung der Attribute jedoch implizit bei der erstmaligen Verwendung. Der folgende Ausschnitt zeigt den Eintrag für die öffentlichen Daten der Personen.

```
'idx.prs.pub' => array(
  'ID' => 3,
  'table' => 'Personen',
  'file' => 'personen.publik.php',
  'label' => 'Öffentliche Daten',
  'help' => 'hilfe.html' )
```

Das Schlüsselfeld `'idx.prs.pub'` besteht aus Gruppen von jeweils drei Buchstaben, die die eindeutige Position der Seite in der Hierarchie festlegen. Die angeführte Seite liegt also auf der dritten Hierarchieebene (`index` → `personen` → `publik`). Jeder Eintrag hat folgende Attribute: Die `ID` zur eindeutigen Kennzeichnung jedes einzelnen Eintrages - unabhängig von der Position in der Hierarchie (Querverweise der Seiten untereinander werden mit Hilfe der `ID`s realisiert, damit sie unabhängig von der Organisation des Baumes funktionieren). Das Feld `table` gibt einen Tabellennamen an. Verknüpfungen werden nur angezeigt, wenn der Benutzer Leseberechtigung auf eines der Felder in der angegebenen Tabelle besitzt. Das `file`-Attribut benennt die zu öffnende php-Datei und `label` gibt die Beschriftung für die Links an. Das `help`-Attribut enthält den Namen der Hilfedatei, die zu dieser Seite angezeigt werden soll.

Farbschema

Die farbliche Gestaltung der Seiten, die Schriftarten und Abstände (Ränder) werden ausschließlich über Style-Klassen definiert. Da alle Styles in einer Datei (`style.css`) zusammengefaßt sind, kann die Oberfläche leicht an ein bevorzugtes Farbschema angepaßt werden.

Das vorliegende Layout mit hellgelben Hintergrundfarben und dunkler Schrift wurde gewählt, um möglichst hohen Kontrast zu erzielen, außerdem belasten helle

Hintergrundfarben die Augen weniger als dunkle. In der Farbenlehre steht helles Gelb für Verstand und Wissen sowie für Heiterkeit und strahlenden Sonnenschein. Die Hintergrundfarbe der Formulare kann zwar ein schlechtes Programm nicht verbessern, sie soll aber den Erfolg der Applikation nicht durch Erzeugung einer ungünstigen Stimmung zerstören...

Verknüpfungen werden in dunklem Rot mit einem Aufhellungseffekt bei Mauskontakt dargestellt, damit sie einfach als Verweis erkannt werden können. Fehlermeldungen werden in einem 'lauten', hellen Rot geschrieben, um Aufsehen zu erregen.

Datenfluß

Bevor im folgenden Abschnitt auf die Klasse `Institut_form` eingegangen werden kann, ist noch eine Erläuterung des Datenflusses nötig:

Den meisten Formularen ist ein Such-Formular vorgelagert, das die wichtigsten Felder der Datensätze in Listenform zum Suchen bereitstellt. Durch Auswahl eines Eintrages in einer solchen Suchliste wird der entsprechende Schlüsselwert in einer registrierten Variablen gespeichert und der Benutzer wird zum Detailformular weitergeleitet. Öffnet ein Benutzer ein Detailformular, so ist also zumeist bereits ein konkreter Datensatz ausgewählt und der Benutzer soll die Daten dieses Datensatzes angezeigt bekommen. In diesem Fall werden also die Steuerelemente mit den Werten aus der Datenbank initialisiert.

Ändert nun der Benutzer Werte im Formular, so wird nach Betätigen der 'Speichern'-Schaltfläche dasselbe Formular nochmals aufgerufen, der neuerliche Aufruf beinhaltet aber in den globalen Variablen die geänderten Werte der Formularelemente, sowie den Namen der gedrückten Schaltfläche. Die geänderten Daten dürfen nun nicht direkt in die Datenbank übernommen werden – sie müssen zuerst die Validierung erfolgreich passieren. Die Validierung der Daten wird durch die `form`-Klasse vorgenommen und verwendet nicht die globalen Variablen, sondern die Steuerelemente (der zu verwendende reguläre Ausdruck ist eine Eigenschaft, Property, jedes einzelnen Steuerelementes). Folglich müssen zuerst die Steuerelemente erstellt und mit den Werten der globalen Variablen initialisiert werden.

Ist die Validierung erfolgreich, werden die Daten in der Datenbank abgespeichert, andernfalls werden Fehlermeldungen zu den falsch ausgefüllten Steuerelementen ausgegeben.

Für die Werte der Steuerelemente gibt es also die in Tabelle 3.3 dargestellten drei Möglichkeiten.

Datenquelle	Anwendungsfall
Datenbank	wenn die Seite ‘normal’, das bedeutet ohne vorherige Betätigung einer Schaltfläche geöffnet wird
globale Variablen	wenn der Datensatz abgespeichert oder ein neuer Datensatz erstellt wird
(leer)/globale Variablen	Bei einem Endlosformular dient die erste Zeile (das erste Formular) als Eingabemöglichkeit für neue Datensätze. Dieses Formular muß daher normalerweise leere Felder enthalten. Nur dann, wenn neu eingegebene Daten die Validierung nicht bestehen, müssen die eingegebenen Werte nochmals angezeigt werden, damit sie korrigiert werden können.

Tabelle 3.3: Werte für Steuerelemente

`Institut_form`-Klasse

Diese Klasse dient als Auffangbehälter für Funktionen, die aus den Formulardateien ausgelagert wurden, um die Quelltexte zu verkürzen und die Übersichtlichkeit zu verbessern. Deshalb besitzt `Institut_form` keinen besonders strukturierten Aufbau, es ist viel mehr als Methodensammlung zu verstehen.

Mit Hilfe von `Institut_form` können Einzel- und Endlosformulare mit geringem Programmieraufwand realisiert werden. Endlosformulare sind Seiten, die mehrere Formulare untereinander enthalten. Diese Art der Darstellung wird unter anderem von den Seiten zur Vergabe der Privilegien verwendet; sie ist jedoch nur dann sinnvoll einzusetzen, wenn die Anzahl angezeigter Formulare sowie die Anzahl von Steuerelementen pro Formular nicht zu groß ist.

Es sind im Laufe der Zeit noch weitere Methoden hinzugefügt worden, die nicht nur im Zusammenhang mit Endlosformularen sinnvoll einzusetzen sind; die Klasse kann also auch in allen anderen Formularen eingesetzt werden.

Die Methoden von `Institut_form` sind relativ kurz gehalten und durchwegs dokumentiert. Hier soll daher nur auf die Zusammenhänge sowie auf mögliche Anwendungsfälle der wichtigsten Methoden eingegangen werden.

`Institut_form():` Dem Konstruktor kann optional eine Abfrage zur Auswahl der Daten übergeben werden. Bei Verwendung mit Endlosformularen wird die Methode `init()` verwendet, um die Anzahl der Datensätze zu ermit-

teln und die Abfrage selbständig auszuführen. Die Anzahl der Datensätze wird benötigt, um die Schleife zur Ausgabe der einzelnen Formulare zu starten.

`choose_content()` wird in Endlosformularen verwendet, um die Initialisierung der Steuerelemente zu vereinfachen. Die Methode retourniert den ersten Parameter, wenn es sich um den leeren Datensatz handelt, sonst den zweiten und vermeidet so eine Anhäufung von `?:`-Operatoren.

`check_delete()` erhält eine Löschabfrage und eine Instanz der Datenbankklasse. Die Methode überprüft, ob Privilegien zum Löschen der in der Property `table` angegebenen Tabelle (– nicht der in der Abfrage verwendeten Tabelle) existieren. Anschließend wird die Abfrage gegebenenfalls ausgeführt. Als Ergebnis retourniert die Methode die Information, ob die Abfrage ausgeführt wurde (Boolean). Diese Methode hat an Bedeutung verloren, da in den meisten Formularen nur noch ein ‘weiches’ Löschen in Form einer `UPDATE`-Anweisung zum Verstecken der Datensätze verwendet wird.

`do_insert()` und `do_update()` haben das gleiche Funktionsprinzip wie `check_delete()`. Wenn die benötigten Privilegien vorhanden sind, führen sie die Abfrage aus und teilen dies dem aufrufenden Programm mit.

`add_buttons()` und `add_deletebutton()` fügen dem übergebenen Formular Schaltflächen zum Neu-Anlegen und Speichern sowie zum Löschen und Wiederherstellen von Datensätzen hinzu.

`show_buttons()` und `show_deletebutton()` zeigen die zuvor hinzugefügten Schaltflächen nun tatsächlich an (starten die Ausgabe des html-Quelltextes). Bei gelöschten Datensätzen wird das Löschdatum und eine Schaltfläche zum Wiederherstellen angezeigt, für alle anderen Datensätze erscheint eine ‘Löschen’-Schaltfläche

`show_delete()` ist trotz seiner Kürze besonders hilfreich: Die Methode erzeugt einen Kriterienausdruck für SQL-Abfragen, der die gelöschten Datensätze ein- oder ausschließt. Bei gelöschten Datensätzen ist im Feld `del` das Datum der Löschung eingetragen, aktive Datensätze haben ein leeres Datum (0) gespeichert. Normalerweise erzeugt `show_delete` ein Kriterium, das nur Datensätze ohne vorhandenem Löschdatum zuläßt. Nur wenn die globale Variable `show_all` gesetzt

ist und der Benutzer das Lösch-Privileg für die angegebene Tabelle besitzt, wird das Kriterium weggelassen und der Benutzer bekommt alle Datensätze geliefert.

`delete_warn()` wird am Beginn von Formularen eingesetzt, um eine Meldung auszugeben, falls der aktuelle Datensatz gelöscht ist. Wenn die Variable `show_all` nicht aktiv ist, wird ein Hinweis ausgegeben, der den Benutzer zum Einstellungen-Formular verweist, wo er `show_all` aktivieren kann. Die Methode retourniert `false` (0), wenn der Datensatz nicht angezeigt werden soll (also wenn er gelöscht ist und `show_all` nicht aktiv).

`start_form()` ruft die Methode `start()` des `form`-Objekts auf. Dadurch wird das öffnende `<form>`-Tag ausgegeben und in weiterer Folge müssen alle Steuerelemente der Reihe nach ausgegeben werden. Um in Endlosformularen die Übersichtlichkeit zu verbessern, kann die Methode an die Ziel-URL (Target) einen Fragment-Anchor anhängen. Dieser Fragment-Anchor wird für jedes Formular vergeben und hat die Form `#A1`, `#A2` usw. Er bewirkt, daß bei langen Endlosformularen der Browser nach dem Abschicken von selbst wieder bis zum aktuellen Formular hinunterrollt. (siehe auch **`add_frag()`**)

`end_form()` ruft die Methode `finish()` des `form`-Objekts auf. Dadurch wird die Ausgabe der Steuerelemente abgeschlossen. Die `form`-Klasse erzeugt html-Code für versteckte Steuerelemente und gibt das schließende `</form>`-Tag aus.

`add_searchlist()` und `show_searchlist()` erzeugen Steuerelemente und JavaScript-Code, um eine ‘clientseitige Suchliste’ zu realisieren. Diese Suchliste wird in leerem Zustand zum Client übertragen und erst dort zur Laufzeit mittels JavaScript gefüllt. Der Vorteil dieser Art der Realisierung ist, daß der Benutzer nach Angabe eines Suchkriteriums nicht auf einen kompletten Neuaufbau der Suchseite zu warten braucht, da die Suche lokal stattfindet. Nachteilig wirkt sich jedoch aus, daß JavaScript für größere Datenmengen nicht geeignet ist, und daß ein kompletter Ausschnitt aus der jeweiligen Tabelle übertragen wird.

Die Methode `add_searchlist()` erzeugt ein Textfeld für die Eingabe eines Suchbegriffes, die Liste für die Suchergebnisse, eine Schaltfläche, um die Suchliste zu aktualisieren sowie eine Schaltfläche, um den Datensatz in einem Detailformular anzeigen zu lassen.

`show_searchlist()` gibt zuerst den benötigten JavaScript-Code aus. Er beinhaltet zwei vorinitialisierte Arrays, die die Schlüssel und die Beschriftung der Listeneinträge enthalten. Die Such-Funktion erstellt aus dem eingegebenen Suchbegriff einen regulären Ausdruck, entfernt alle (bis auf die ersten zwei[4]) Listeneinträge aus der Suchliste und prüft anschließend jeden Wert aus dem Daten-Array. Die ‘passenden’ Einträge werden wieder in die Liste aufgenommen und stehen zur Auswahl bereit.

Tests haben ergeben, daß diese Realisierung bereits bei etwa 1000 Datensätzen inakzeptabel hohe Wartezeiten und auch unvorhersehbares Verhalten verursachen kann. Deshalb mußte beim Formular zur Schlüssel-Eingabe ein serverseitiges Suchformular verwendet werden.

3.2.4 Spezielle Formulare

Dieser Abschnitt bietet einen Überblick über die internen Abläufe in den Formularen und geht weiters auf die Besonderheiten spezieller Formulare - wie dem Suchformular - ein.

Einzel- und Endlosformulare

Bei der Entwicklung der Formulare wird hauptsächlich die Methode `inhalt()` für die von `Site` abgeleitete Klasse implementiert. Es wird also nur jener Programmteil neu implementiert, der die Inhalts-Zelle der Gliederungstabelle ausfüllt; alle weiteren Vorgänge können aufgrund der starken Modularisierung unberücksichtigt bleiben.

Einzel- und Endlosformulare haben viele Gemeinsamkeiten im Aufbau. Deshalb wird der Programmablauf hier für beide Arten gemeinsam beschrieben und auf die Unterschiede explizit hingewiesen.

Am Beginn von `inhalt()` wird überprüft, ob der Benutzer die nötigen Privilegien besitzt, um dieses Formular öffnen zu dürfen. Benutzer ohne Leserecht bekommen nur eine Fehlermeldung präsentiert.

Anschließend werden die benötigten Klassen instantiiert. Datenbankzugriffsvariablen werden möglichst für die gesamte Laufzeit der Methode weiterverwendet, da bei manchen Datenbankservern (allerdings nicht bei MySQL) der Verbindungsaufbau besonders lange dauert und daher ‘sparsam’ mit Datenbankverbin-

[4]In der ersten Zeile steht “Bitte wählen. . . ”, in der zweiten Zeile steht der voreingestellte Wert, falls die Liste zur Auswahl eines Attributes, etwa einer verantwortlichen Person, verwendet wird.

dungen umgegangen werden sollte. Deshalb werden auch möglichst Referenzen auf eine bestehende Datenbankinstanz als Parameter an Hilfsfunktionen übergeben, anstatt in den Funktionen neue Instanzen zu erstellen. In den Formularen wird die Instanz `$db` immer als jene Instanz initialisiert, die die anzuzeigenden Daten enthält. Für weitere Abfragen werden Instanzen `$db1`, `$db2` usw. verwendet.

Da die Steuerelemente zum Löschen nicht initialisiert sein müssen, wird zuerst die Lösch- und Wiederherstellungsprozedur angewandt. Dabei wird zuerst überprüft, ob die notwendigen Abhängigkeiten erfüllt sind (siehe Anhang B). Wenn durch das Löschen eine Anomalie erzeugt würde, wird statt dessen eine Fehlermeldung mit der Liste der störenden Datensätze angezeigt. Wenn alle Abhängigkeiten erfüllt sind, wird tatsächlich gelöscht bzw. wiederhergestellt. Datensätze werden gelöscht, indem das aktuelle Datum in das Attribut `del` eingetragen wird. Beim Wiederherstellen wird das Löschdatum mit dem Wert 0 überschrieben.

Bevor anschließend eventuell geänderte Daten in die Datenbank übernommen werden, müssen zuerst die Steuerelemente initialisiert werden, um die Daten validieren zu können. Das stellt beim Einzelformular kein Problem dar – die Steuerelemente müssen in jedem Fall initialisiert werden. Bei Endlosformularen jedoch müssen die Steuerelemente für jedes einzelne Formular (in der später folgenden Schleife) und hier - am Beginn - erzeugt werden. Deshalb wird bei Endlosformularen die Erstellung der Steuerelemente in eine neue Methode (`prepare_form()`) ausgelagert. Wenn die Validierung nicht erfolgreich abgeschlossen werden kann, retourniert die Methode `form->validate()` alle gefundenen Fehlermeldungen als Zeichenkette. Einzelformulare geben diese Fehlermeldung sofort aus; Endlosformulare geben die Meldung innerhalb der Schleife erst bei dem betreffenden Formular aus.

Wenn die Validierung bestanden wurde, so werden (je nach gedrückter Schaltfläche) ein INSERT- oder ein UPDATE-Statement zusammengestellt und ausgeführt, falls die nötigen Erstellungs- bzw. Schreibrechte existieren.

Nun werden die Steuerelemente ausgegeben. Bei Einzelformularen kann dies direkt passieren; bei Endlosformularen müssen alle vorhandenen Datensätze in einer Schleife durchlaufen und für jeden Datensatz ein eigenes Formular erzeugt werden. Für die Ausgabe wird meist eine tabellarische Form gewählt, damit die Steuerelemente trotz unterschiedlich langer Beschriftungen stets untereinander stehen.

Bei manchen Formularen werden am Ende die verknüpften Datensätze in Listenfeldern angezeigt (z.B. bei den Räumen die Klappen und die Personen, die diesem Raum benutzen), um die Übersicht zu verbessern.

Such-Formular

In der bisher verwendeten Datenbank war es den Benutzern möglich, selbst Abfragen zu erstellen, um Datensätze aufgrund von Bedingungen auszufiltern, mit anderen Datensätzen zu verknüpfen und die Ergebnisse nach ausgewählten Feldern zu sortieren. In Microsoft Access werden jedoch alle Abfragen in einem gemeinsamen Namensraum gespeichert. Dadurch werden die Abfragen aller Benutzer miteinander vermengt, was dazu führt, daß Abfragen nicht mehr wiedergefunden und deshalb mehrmals neu erstellt werden.

Daher sind die Ziele für das neu zu erstellende Such-Formular auf Benutzerfreundlichkeit und Schaffung eigener Namensräume für die Abfragen der einzelnen Benutzer ausgerichtet. Um die Erstellung der Abfragen zu vereinfachen, werden die Verknüpfungsausdrücke (‘`JOIN`’) in unterschiedlichen Setups vordefiniert. Der Benutzer wählt also einen ‘Ausschnitt’ der Datenbank und definiert Kriterien für die darin enthaltenen Felder.

Abfragen werden getrennt nach Benutzern gespeichert. Es soll jedoch trotzdem möglich sein, Abfragen für die Allgemeinheit freizugeben, wobei nur der Besitzer einer Abfrage die Möglichkeit hat, diese zu verändern (speichert ein Benutzer die Abfrage eines anderen Benutzers in veränderter Form ab, so wird eine Kopie im Namensraum des neuen Besitzers erstellt; das Original bleibt davon unberührt).

Zum Erstellen der Datenabfrage werden die Parameterwerte aus dem Such-Formular verwendet. Auch beim Abspeichern von Abfragen werden nur die Variablen und nicht die fertige SQL-Abfrage gesichert, weil aus der fertigen Abfrage nur unter großem Aufwand wieder auf die eingestellten Parameterwerte rückgeschlossen werden kann.

Die Erstellung der SQL-Abfragen aus den Variablen wird in einer eigenen Klasse implementiert und in die Datei `query_setup.inc` ausgelagert. Aufgabe der Klasse ist die Auswertung der Formular-Variablen, Erstellung einer internen Repräsentation aus den Variablen sowie Erstellung der SQL-Abfrage.

Im Such-Formular wählt der Benutzer `FeldIDs`, wenn er Datenfelder für Kriterien oder zum Anzeigen selektiert. Aus den (numerischen) `FeldIDs` können in der `Felder`-Tabelle die vollständigen Feldnamen und die Beschriftungen ermittelt werden. Die `FeldIDs` werden aber auch zur Auswahl der Attribute verwendet (die Bitfelder, denen in der `Attribute`-Tabelle erst zur Laufzeit Bezeichnungen zugeordnet werden). Attribute werden in der Form ’Tabellenname.Stellenwert’ dar-

gestellt. Außerdem werden negative `FeldID`s verwendet, um Felder darzustellen, die in der `Felder`-Tabelle in dieser Form nicht existieren, wie zum Beispiel den Betreuer einer Tätigkeit. Die Funktion `hole_feldname()` übersetzt die `FeldID`s in die vollständigen Feldnamen, ermittelt, ob das Leseprivileg für die Felder vorhanden ist und retourniert einen kompletten Feldinfo-Datensatz als assoziatives Array.

Die Methode `init()` verwendet die Formularvariablen, um die interne Repräsentation der Abfrage aufzubauen. Diese besteht aus einem Profil, das den Verknüpfungsausdruck festlegt, der Liste der anzuzeigenden Felder, einer Liste von Kriterien und einer Liste von Feldern, die zum Sortieren verwendet werden sollen.

Für die Kriterien- und Auswahlfelder zum Sortieren wird im Formular jeweils um eine Zeile mehr erstellt, als momentan benötigt wird. Dadurch erscheint nach Ausfüllen einer Zeile beim nächsten Anzeigen eine neue Zeile für die nächsten Werte. Zeilen, in denen keine gültige `FeldID` ausgewählt sind, werden entfernt. Wenn das Leserecht für ausgewählte Felder nicht vorhanden sind, werden diese Felder einfach ignoriert und nicht angezeigt.

Zur Erzeugung der SQL-Abfrage dienen die in den folgenden Absätzen beschriebenen Methoden. `create_selectpart()` erzeugt den `SELECT`-Teil der Abfrage und die Überschriftenzeile für die Ergebnistabelle. Das Schlüsselwort `DISTINCTROW` ist in der Abfrage unbedingt notwendig, weil durch den für die meisten Abfragen unnötig ausführlichen Verknüpfungsausdruck (es werden nicht immer aus allen Tabellen im Verknüpfungsausdruck Felder angezeigt) redundante Zeilen im Ergebnis erzeugt werden.

`create_frompart()` setzt die vordefinierten Suchprofile in Verknüpfungsausdrücke um. Diese werden als Kette von `LEFT JOIN`-Konstrukten aufgebaut und beginnen mit der Tabelle, nach der das Suchprofil benannt ist. So wird sichergestellt, daß alle Datensätze der 'Haupttabelle' im Ergebnis enthalten sind. Alle weiteren Felder enthalten nur dann Werte, wenn verknüpfte Datensätze vorhanden sind. Würde man `INNER JOIN` als Verknüpfung verwenden, so würden nur jene Datensätze im Ergebnis enthalten sein, die Beziehungen zu *allen* Tabellen im Verknüpfungsausdruck besitzen, was mit zunehmender Anzahl verknüpfter Tabellen immer unwahrscheinlicher wird.

`create_wherepart()` erstellt die Liste der Kriterien. Zu den benutzerdefinierten Kriterien müssen noch die zum Ausschluß der gelöschten Datensätze hinzugefügt werden (Diese Bedingungen hätten am besten in die `FROM`-Abschnitten gepaßt. Dort dürfen aber nur an der Verknüpfung beteiligte Felder verwendet wer-

den). Die benutzerdefinierten Kriterien werden an die Ausschlußbedingungen angeschlossen. Sie bestehen aus jeweils einer Vergleichsfunktion, dem Feldnamen, der optionalen Negation, sowie der Verbindung (AND/OR) zur nächsten Bedingung.

`create_orderpart()` gibt schließlich die Sortierbedingungen aus. Bereits in der Methode `init()` wurde sichergestellt, daß alle Felder, nach denen sortiert werden soll, auch in der Ergebnistabelle enthalten sind.

Die Ausgabe der Daten erfolgt normalerweise als html-Tabelle. Zum Exportieren der Daten ist aber auch eine durch Tabulatoren getrennte und eine durch Komma getrennte Listenform vorgesehen.

Organigramm

Die Datenbank speichert die Organisationsstruktur des Instituts in einer hierarchischen Struktur als Baum. Die Baumstruktur wird auf die Datenbank abgebildet, indem jede Organisationseinheit mehrere ihr untergeordnete Organisationseinheiten haben kann. Organisationseinheiten können Abteilungen, Arbeitsgruppen oder ganze Institute sein; es können also mehrere unabhängige Teilbäume realisiert werden. Auch die Benennungen der Typen von Organisationseinheiten können in einer eigenen Tabelle definiert werden und sind somit an die Gegebenheiten anzupassen.

Das Organigramm-Formular stellt die Hierarchie mit Hilfe von Tabellen als Baum dar und kann dazu benutzt werden, die Hierarchie zu überprüfen und die Angehörigen jeder Abteilung aufzulisten.

Das Programm zur Erstellung des Organigramms erzeugt zunächst aus der Tabelle der Organisationseinheiten eine Baumstruktur im Speicher. Darin sind die obersten (Wurzel-) Einheiten und alle untergeordneten Organisationseinheiten vermerkt. Dieser Baum muß nun in eine Tabelle übersetzt werden, damit er als html-Seite ausgegeben werden kann.

html-Tabellen sind vergleichbar mit Matrizen und müssen zeilenweise an den Browser ausgegeben werden. Da Baumstrukturen am besten Astweise rekursiv durchlaufen werden können, wird zunächst in einem eigenen Durchlauf der Baum Astweise durchwandert und in ein zweidimensionales Array (eine Matrix) umkopiert. Dieser Zwischenschritt erweist sich als sehr hilfreich, da in Arrays wahlfrei geschrieben werden kann.

Die erzeugte Matrix wird anschließend zeilenweise in html-Tabellenelemente übersetzt und so an den Browser ausgegeben. Abhängig von den Einstellungen im Formular werden zu jeder Organisationseinheit alle bzw. nur die internen Angehörigen mit ausgegeben.

Browser-spezifische Funktionen

Abseits der Kernfunktionalität der Webapplikation sind noch weitere Funktionen zu implementieren, um auf 'Eigenheiten' bestimmter Browser einzugehen.

Schutz gegen unbeabsichtigten Formular-Submit Formulare dürfen nur genau *ein einziges Mal* abgeschickt werden können, und zwar erst dann, wenn sie vom Browser vollständig geladen worden sind. Zum einen gibt es Benutzer, die doppelt auf Schaltflächen klicken, zum anderen führt der Browser JavaScripts möglicherweise nicht aus, wenn sie im Quelltext erst später positioniert sind und daher noch nicht heruntergeladen wurden. Daher wird eine Sperre eingebaut, die das Abschicken des Formulars erst nach vollständigem Laden ermöglicht und nur genau *ein* Submit zuläßt.

Dazu wird eine globale JavaScript-Variable (`ready`) mit 0 (false) initialisiert. Erst mit Eintritt des Ereignisses `onLoad` wird die Variable freigegeben (auf 1 gesetzt). Bevor nun ein Formular abgeschickt wird, prüft eine JavaScript-Funktion, ob `ready` freigeschaltet ist. Dazu wird die "after-validation" Funktion der Form-Klasse verwendet (siehe `myform->get_start()` in `local.inc`). Wenn die Sperrvariable nicht aktiv ist, wird der Submit abgebrochen.

Die Sperrfreigabe wird auch für die Suchlistenformulare eingesetzt, um eine besondere Eigenheit zu vermeiden: Das Drücken der Return-Taste wird von manchen Browsern als Betätigung des ersten Submitbuttons interpretiert. Das wäre in den vorgelagerten Suchformularen besonders unerwünscht, da Benutzer nach Eingabe eines Suchbegriffes 'instinktiv' Return drücken und so auf die leere Detailseite weitergeleitet werden, da sie noch keinen Eintrag in der Liste der Suchergebnisse ausgewählt haben. Deshalb wird in Suchformularen die Submitsperre nicht bereits beim Laden des Formulars aufgehoben, sondern erst durch Klicken auf die Suchen-Schaltfläche.

Browseridentifikation und JavaScript-Erkennung: Die Web-Applikation verwendet JavaScript für die Challenge-Response Authentifizierung, für die 'clientseitigen Suchformulare', zum Schutz gegen unbeabsichtigtes Abschicken

von Formularen und zum Überwachen von Steuerelementen, auf die der Benutzer keine Editierrechte besitzt. Die Validierung der Daten erfolgt ausschließlich am Server, da mit JavaScript erzeugte Meldungsfenster den Arbeitsablauf des Benutzers stören und die serverseitige Validierung zuverlässiger und flexibler zu realisieren ist. Darüber hinaus wäre es ein Sicherheitsrisiko, sich auf die Ausführung von Validierungsscripts zu verlassen, da 'bösartige' Benutzer die Validierungsroutinen aus dem Seitenquelltext entfernen und so ungültige Daten in die Datenbank einschleusen könnten.

Ein Browser, der JavaScripts nicht ausführt, kann die Anwendung nicht sinnvoll einsetzen, weil die Suche nach Datensätzen nicht funktioniert. Deshalb werden im Zuge der Authentifizierung Benutzer ausgeschlossen, deren Browser JavaScript nicht verarbeitet. Die JavaScript-Erkennung ist nur über Umwege realisierbar und wird hier ideal mit der erhöhten Sicherheit der verschlüsselten Authentifizierung verkoppelt: Wenn der Wert der Response bei der Anmeldung leer ist, so hat der Browser das Script nicht ausgeführt und kann die Anmeldung nicht passieren.

Da noch immer ältere Browser verwendet werden, die nicht vollständig kompatibel zu JavaScript, Stylesheets und html 4 sind, wurde auch eine Versionsüberprüfung für den Browser eingebaut. Im Zuge der Anmeldung wird die Browserkennung ('user agent') ausgewertet. Die Prüfung erfordert, daß sich der Browser als 'Mozilla' identifiziert und als Versionsnummer zumindest 4 bzw. Opera ab Version 5 angibt. Dadurch werden die Browser von Netscape und Microsoft ab Version 4 bzw. Opera ab 5 zugelassen. Für weniger bekannte Browser muß zunächst durch einen Testlauf sichergestellt werden, daß die Darstellung korrekt erfolgt und Skripte ordnungsgemäß ausgeführt werden, bevor die Prüfung auf andere Browsernamen und Versionen erweitert werden kann (siehe `auth->auth_validatelogin()` in `local.inc`).

3.3 Installation

Bei der Installation der Institutsverwaltung muß die Datenbank mit den aktuellen Datenständen initialisiert und der Webserver für die Abarbeitung der php-Skripte konfiguriert werden. Bei den Einstellungen spielen Datensicherheit und Schutz gegen Angriffen auf die Infrastruktur eine gewichtige Rolle.

Übergabe der Datenbestände

Um die Daten aus der bestehenden Datenbank zu exportieren, wurde eigens ein Export-Frontend entworfen. Diese Datenbankanwendung erzeugt mit den bestehenden Daten SQL-Anweisungen zum Einfügen der Datensätze in die neue Datenbankstruktur. Dabei werden die Daten an das geänderte Datenbankschema angepaßt und so formatiert, daß sie von MySQL korrekt weiterverarbeitet werden können. Als Ergebnis erhält man eine Textdatei, die alle Einfügeanweisungen enthält.

Nach dem Einspielen der Daten in den Datenbankserver müssen fehlende Informationen manuell ergänzt werden. Dazu zählen alle Daten, die erst im neuen Datenbankmodell hinzugefügt wurden (z.B. Projekte und Computer) oder aus anderen Gründen noch nicht eingegeben wurden.

Einrichten von MySQL

In Kapitel 6 des MySQL-Handbuches [6] wird das Privilegiensystem von MySQL beschrieben. Das Studium der Original-Lektüre wird dem Leser empfohlen; hier kann nur eine kurze Zusammenfassung wiedergegeben werden.

Bei der Vergabe von Zugriffsrechten muß der Grundsatz befolgt werden, niemandem mehr Rechte zu geben, als unbedingt nötig. Der Datenbankserver (`mysqld`-Prozeß) muß unter einem Account (Linux Username) gestartet werden, der ausschließlich und als einziger auf das Verzeichnis zugreifen darf, das die Tabellen enthält. Für den Webserver muß ein MySQL-Benutzer eingerichtet werden, der ebenfalls nur mit minimalen Privilegien ausgestattet werden darf[5]. Der MySQL-Benutzer darf auf die Tabellen der Institutsverwaltung lediglich `SELECT`-, `INSERT`-, `UPDATE`- und `DELETE`-Anweisungen anwenden. Zusätzlich benötigt er dieselben Rechte für die Benutzertabelle, die in der IDM-Datenbank liegt. Zugriffe auf andere Datenbanken – vor allem auf die `mysql`-Datenbank müssen unbedingt verboten werden. Alle weiteren Privilegien, insbesondere zum Ändern der Tabellenentwürfe (`CREATE`, `ALTER` und `DROP`) oder zum Zugriff auf Dateien (`LOAD DATA INFILE '/etc/passwd' INTO TABLE GeklautePasswoerter;`) müssen ihm verwehrt bleiben, da sie in der Web-Applikation nicht verwendet werden.

[5]Die Benutzer der Web-Applikation authentifizieren sich gegenüber dem Webserver - nicht gegenüber MySQL. Die Web-Applikation hat alle möglichen Rechte und bestimmt selbst, welche Rechte sie an jeden einzelnen Benutzer weitergibt.

Sofern der Webserver auf derselben Maschine läuft wie MySQL, sollte für die IP-Adresse des Benutzers `127.0.0.1` angegeben werden, damit nur vom Server aus auf die Datenbank zugegriffen werden kann. Selbstverständlich darf in MySQL kein einziger Benutzer ohne Paßwort eingerichtet sein, um unbefugte Zugriffe zu verhindern. Für zusätzlichen Schutz kann das Paßwort verschlüsselt abgespeichert werden (`ENCRYPT()`-Funktion).

In der `user`-Tabelle wird also ein Benutzer angelegt, der vorerst noch kein einziges Privileg besitzt und als Host die IP-Adresse des Webservers eingegeben hat. Diesem unprivilegierten Benutzer werden anschließend durch zwei Einträge in der `DB`-Tabelle die Zugriffe auf `Institut` und `IDM` erlaubt. So kann der User nicht auf die anderen Datenbanken zugreifen.

Schützen der Zugangsdaten

Die Zugangsdaten zur Anmeldung am Datenbankserver sind im Programm im Klartext enthalten und dürfen unter keinen Umständen nach außen gelangen, weil ein Benutzer mit Hilfe der Zugangsdaten volle Änderungs- und Löschprivilegien erhält.

Unter 'normalen' Umständen sind die Zugangsdaten im Quelltext gut geschützt, weil der Benutzer stets nur den durch php interpretierten Quelltext bekommt, der die Zugangsdaten natürlich nicht mehr enthält. Sollte jedoch aus irgendeinem Grund der php-Interpreter nicht korrekt ausgeführt werden, so würde ein Benutzer den Quelltext erhalten und könnte die Zugangsdaten mißbrauchen. Dieser Fall könnte durch eine Fehlkonfiguration, durch einen Programmfehler im php-Modul oder 'absichtlich' durch einen Angriff von außen herbeigeführt werden. Um die Zugangsdaten gegen diese Angriffsmethode zu schützen, müssen sie in eine Datei ausgelagert und per `include()` in die Webseiten eingebunden werden. Die Datei muß in einem Verzeichnis liegen, das außerhalb des Dokumentenverzeichnisses des Webservers liegt. Dadurch kann der Angreifer die Datei nicht einfach durch Eingabe des Namens im Browser abrufen. Die Zugangsdaten werden also entweder normal verwendet oder erst gar nicht eingebunden, wenn php nicht läuft. In beiden Fällen gelangen sie nicht zum Client.

Im Fall der Institutsdatenbank sind die Zugangsdaten zentral in der Klassendefinition von `DB_Institut` (`local.inc`) abgelegt. Diese Datei sollte gemeinsam mit allen anderen PHPLIB-Dateien nicht im Document Root (z.B. `/usr/local/httpd/htdocs`) erreichbar sein, sondern in einem eigenen Zweig; es empfiehlt sich ein Pfad wie z.B. `/usr/local/httpd/php`).

Die Quelltexte der Applikation müssen jedenfalls auch auf Datei-Ebene durch das Betriebssystem gesichert werden. Außer dem Administrator und dem Webserver sollte niemand die Quelltexte lesen dürfen. Alle Linux User müssen durch Paßwörter geschützt sein und per FTP sollte kein Zugriff auf die Quelltext-Verzeichnisse möglich sein – auch nicht zum Hochladen der Seiten.

Einrichten von Apache

Der Apache Webserver darf keinesfalls als `root`-User ausgeführt werden, da er sonst alle erdenklichen Rechte besitzen würde. Im Falle eines Angriffes könnte jemand einen Totalschaden anrichten, nachdem er den `httpd` eingenommen hat. In der Praxis wird daher zumeist ein eigener Benutzername eingerichtet, der minimale Privilegien besitzt (`wwwrun` oder `nobody`).

Falls am Server CGI-Programme verwendet werden sollen, so ist es sehr empfehlenswert, alle CGI-Programme in ein gemeinsames Verzeichnis zu legen, auf das nur vertrauenswürdige Benutzer Schreibrechte besitzen. Dadurch kann der Administrator kontrollieren, welche Programme als CGI ausgeführt werden dürfen (‘script aliased CGI’). Die Institutsdatenbank führt selbst keine weiteren Programme aus. Falls php als Modul eingesetzt wird, kann CGI auch vollständig deaktiviert werden.

Zur Sicherheit sollte für den gesamten Server die Ausgabe von Verzeichnislistings verboten werden. Die Verzeichnislistings enthalten für Angreifer interessante Informationen über den Server und über die vorhandenen Dateien. Um Listings abzuschalten wird die `Options`-Direktive im `<Directory>`-Abschnitt von `httpd.conf` verwendet:

```
<Directory /usr/local/httpd/htdocs>
  ...
  Options -Indexes [...]
</Directory>
```

php: Die Anwendung kann - und soll - im ‘php Safe Mode’ ausgeführt werden. Dadurch wird sichergestellt, daß ein Skript nur Zugriff auf jene Dateien erhält, die denselben Eigentümer wie das Skript selbst haben. In der aktuellen Form greift die Anwendung ausschließlich auf den Datenbankserver zu; es werden keine externen Dateien angelegt oder gelesen (ausgenommen natürlich die Skripte selbst).

Für die höchstens erlaubte Ausführungszeit (`max_execution_time`) sollte ein ‘vernünftiger’ Wert eingestellt werden, damit Skripte nicht in Endlosschlei-

fen hängen bleiben können (jedenfalls weniger als 30 Sekunden). Der erlaubte Speicherbedarf für Skripte sollte ausreichend bemessen werden, da sonst komplexe Suchanfragen nicht funktionieren könnten. Werte von 8MB bis 16MB sollten jedenfalls ausreichend sein.

Die Funktion `register_globals` muß unbedingt aktiviert werden, weil Formularvariablen unabhängig von ihrer Herkunft (GET, POST, Cookie) im Programm einfach als globale Variablen angesprochen werden.

Die Option `auto_prepend_file` kann verwendet werden, um `prepend.php3` automatisch einzubinden. Diese Datei ist Teil von PHPLIB und wird benötigt, um weitere PHPLIB-Dateien einzubinden und Konstanten zu definieren. Sollte die Verwendung dieser Direktive nicht möglich sein, so kann `prepend.php3` auch per `include`-Befehl in der Datei `pagestart.php` eingebunden werden, weil diese in allen Dateien verwendet wird.

Die Direktive `include_path` sollte den Pfad zu den PHPLIB-Dateien enthalten. Als Alternative kann auch die Konstante `$_PHPLIB['libdir']` verwendet werden, die in `prepend.php3` definiert wird.

SSL: Eine sehr empfehlenswerte Einführung in das Thema Verschlüsselung bietet die Dokumentation zu `mod_ssl` [18], dem Verschlüsselungsmodul für Apache. Das Grundprinzip der Verschlüsselung ist, daß bestimmte mathematische Funktionen nach bisherigem Wissensstand nur unter großem Aufwand umkehrbar sind. Zum Beispiel die Aufspaltung extrem großer Zahlen in ihre Primfaktoren, wenn sie aus genau zwei Primfaktoren bestehen[6]. Als besonders sicher und gleichzeitig einfach gilt die Bildung einer Art Ziffernsumme, aus der sich der Originaltext nicht ermitteln läßt, die aber auch für geringfügig veränderte Originaltexte bereits möglichst stark abweichende Ergebnisse liefert. Damit können Datenübertragungsfehler und absichtlich eingefügte Manipulationen erkannt werden.

Mit Hilfe solcher Funktionen kann man zwei wichtige Effekte erzielen:

- Man kann Nachrichten *verschlüsseln*. Der Nachrichtentext wird mit dem Public Key, der öffentlich zugänglich ist, einem Verschlüsselungsalgorithmus unterworfen. Der entstandene Ciphertext kann nur von dem Benutzer wieder lesbar gemacht werden, der den geheimen Private Key kennt.

- Man kann *verifizieren*, daß eine Nachricht unverfälscht ist und tatsächlich von dem Absender stammt, der angegeben ist (digitale Signatur). Dazu

[6] Es ist nicht bewiesen, daß kein effizientes Verfahren zum Aufspalten langer Zahlen existiert – der Öffentlichkeit ist jedoch kein Verfahren bekannt.

wird die Prüfsumme der Nachricht (das Message Digest) mit dem Privatschlüssel des Absenders verschlüsselt: Jeder kann die digitale Signatur mit dem öffentlichen Schlüssel des Absenders lesen, aber nur, wer den Privatschlüssel kennt, kann die digitale Signatur korrekt erstellen; so ist die Authentizität der Nachricht sichergestellt, wenn das Schlüsselpaar tatsächlich dem Absender gehört.

Da die Erzeugung von neuen Schlüsselpaaren kein Problem darstellt, benötigt man eine vertrauenswürdige Stelle (Certificate Authority), die glaubhaft bestätigen kann, daß ein bestimmter öffentlicher Schlüssel tatsächlich jener Person gehört, die im Zertifikat angegeben ist. Im Zuge der Registrierung eines Schlüsselpaares wird die Identität des Antragstellers nachgeprüft und ein Zertifikat erstellt, das den Namen und den öffentlichen Schlüssel der Person sowie den Namen der Zertifizierungsstelle enthält. Das Zertifikat wird mit dem privaten Schlüssel der Zertifizierungsstelle signiert. Damit ist sichergestellt, daß nur die Zertifizierungsstelle das Zertifikat ausstellen kann. Gängige Browser kennen die öffentlichen Schlüssel der größeren Zertifizierungsstellen und können so Zertifikate öffnen.

Bei der Verwendung von SSL zur Verschlüsselung von http-Sitzungen wird das SSL-Protokoll zwischen TCP/IP und http eingeschaltet. Es werden also http-Pakete in SSL-Rahmen verpackt und diese mit TCP/IP versendet.

Da die Anwendung des oben beschriebenen asymmetrischen Verfahrens zu sehr rechenintensiv wäre, wird dieses Verfahren nur dazu benützt, um die Identitäten der Kommunikationspartner zu überprüfen und anschließend ein 'gemeinsames Geheimnis' (shared secret; Schlüssel für ein symmetrisches Verschlüsselungsverfahren, den nur die beiden Kommunikationspartner kennen) ausgehandelt.

Zumeist wird ohnehin nur die Identität des Servers auf diese Art überprüft, damit der Benutzer weiß, daß er mit dem 'echten' Server kommuniziert. Die Identität des Benutzers wird durch Vergabe von Namen und Paßwörtern auf einer höheren Ebene überprüft.

Die Installation von SSL mit Apache erfolgt am besten streng nach Anleitung. Zunächst muß Apache, `mod_ssl` und OpenSSL installiert sein. Mit OpenSSL kann ein Schlüsselpaar erstellt werden, das in Form eines Certificate Signing Request (CSR) an eine Zertifizierungsstelle übergeben wird. Man erhält ein signiertes Zertifikat, mit dem sich der Server als authentisch ausweisen kann. Man

kann das Zertifikat zum Testen auch selbst (mit dem eigenen soeben erzeugten Private Key) signieren; Internet Explorer akzeptiert selbstsignierte Zertifikate jedoch nicht. Für die tatsächliche Anwendung ist also ein 'echtes' Zertifikat unbedingt notwendig.

Das Zertifikat und der private Schlüssel werden nun auf den Server kopiert. Die Schlüsseldatei sollte nur von root und dem Webserver lesbar sein. Sie wird zusätzlich verschlüsselt abgespeichert und beim Starten des Serverprozesses ist die Eingabe einer Passphrase notwendig, damit der private Schlüssel gelesen werden kann. Für sonst gut abgesicherte Umgebungen kann auf diese zusätzliche Sicherheit verzichtet werden, um einen unbeaufsichtigten Start des Servers zu ermöglichen.

In der Konfiguration (`httpd.conf`) kann eingestellt werden, für welche Domänen welche Verschlüsselung zu verwenden ist. Dazu sind in der Dokumentation bereits einige Beispiele aufgeführt.

3.4 Wartung

Im laufenden Betrieb ist es notwendig, den zu Beginn gesetzten Qualitätsstandard der gespeicherten Informationen aufrecht zu erhalten. Dazu dienen zunächst die laufend durchgeführten Überprüfungen, etwa vor dem Löschen oder Wiederherstellen von Datensätzen. Zusätzlich gibt es ein Formular, das verwaiste Datensätze anzeigt. Hier werden sämtliche Beziehungen, die im EER-Diagramm (2.1) definiert sind, überprüft.

Für alle Beziehungen wird überprüft, ob nicht-gelöschte Datensätze mit gelöschten Datensätzen in Beziehung stehen. Es darf zum Beispiel keine Etage geben, die nicht zu einem Gebäude gehört, weil eine Etage für sich alleine nicht sinnvoll ist. Außerdem wird überprüft, ob verknüpfte Datensätze überhaupt existieren (wenn also einer Etage ein Gebäudedatensatz zugeordnet wäre, der nicht mehr existiert).

Löschanomalien (übergeordneter Datensatz als gelöscht markiert) können direkt mit Hilfe der Oberfläche repariert werden, indem entweder der untergeordnete Datensatz gelöscht oder der Übergeordnete wiederhergestellt wird. Bei Verweisen, die ins Leere zeigen, sollte unbedingt der Administrator verständigt werden, da solche Zustände nicht vorkommen dürfen. Mit einem Administrationstool für Datenbanken kann dann ein genauerer Einblick in die tatsächlichen Datenstrukturen vorgenommen werden. Da über die Oberfläche keine Datensätze der im

EER-Diagramm 2.1 dargestellten Tabellen tatsächlich gelöscht werden, sind echt verwaiste Datensätze ein Alarmsignal zur Erkennung von Eindringlingen oder Systemfehlern.

An dieser Stelle soll auch auf die Notwendigkeit von Datensicherung (Backups) hingewiesen werden. Im schlimmsten Fall müssen alle Änderungen an der Datenbank seit dem letzten Backup wiederholt werden, wenn ein Totalausfall erlitten wurde[7]. Die Periodendauer zwischen den Backups ist also genau die Zeitspanne, die mit vertretbarem Aufwand nachgearbeitet werden kann.

Neben spezialisierten Lösungen für Server, die den kompletten Inhalt der Festplatten auf Band sichern, gibt es in MySQL auch eine sehr einfache Möglichkeit, komplette Datenbanken in SQL-Anweisungen zum Erstellen der Strukturen und zum Einfügen der Daten zu exportieren.

Das Kommando

```
mysqldump -A -c > dump.sql
```

erzeugt eine Textdatei, die Anweisungen enthält, um *alle* vorhandenen Datenbanken zu erstellen und mit den vorhandenen Werten zu füllen. Diese Datei kann per Skript regelmäßig erstellt und über eine verschlüsselte Verbindung auf einen FTP-Server kopiert werden, damit sie auf einer geographisch entfernten Maschine abgelegt werden kann.

Um die Datenbanken wiederherzustellen, führt man einfach das Kommando

```
mysql < dump.sql
```

aus, wenn man eine aktuelle Sicherungsdatei zur Hand hat...

[7]Während sich der Finger in Richtung Enter-Taste bewegt, um das neue Backup zu starten, schlägt der Blitz ein und ein unangenehmer aber eindeutiger Geruch dringt aus dem Servergehäuse...

3.5 Zusammenfassung

Die neue Institutsdatenbank soll die Verwaltung vereinfachen und bestmöglich unterstützen. Durch Plattformunabhängigkeit und Verwendung standardisierter Sprachen und Protokolle bleibt der Weg offen, die Datenbank in der Zukunft auf einer anderen Serverhardware mit einer anderen Serversoftware weiterverwenden zu können. Um die Lösung möglichst langlebig zu gestalten, wurde an mehreren Stellen versucht, etwas mehr als nur den aktuell geforderten Leistungsumfang zu implementieren - etwa bei den Attributen.

Um der neuen Datenbank überhaupt die Chance zu geben, von den Benutzern angenommen zu werden, sind große Anstrengungen unternommen worden, um ein einheitliches Layout mit konsistenten Navigationshilfen und einer ansprechenden Farbgebung zu erzielen. Bei der Entwicklung der Seiten wurde stets versucht, den Blickwinkel des Benutzers einzunehmen, um möglichst einfache und klare Abläufe zu erreichen.

Verwaltungssoftware muß regelmäßig angepaßt werden, um neuen Anforderungen gerecht zu werden, die häufig von außen vorgegeben sind. Erweiterungen werden durch den modularen Aufbau und den erreichten hohen Grad an Abstraktion besonders unterstützt. Die Formulare besitzen einen einheitlichen Aufbau; deshalb kann beim Entwurf einer neuen Seite ein bestehendes Formular als Vorlage verwendet werden.

In der aktuellen Form ist die Institutsdatenbank nunmehr auf ein neues, stabiles Fundament gestellt worden, das ausbaufähig ist. Die erste Phase der Anwendung der Applikation wird zeigen, welche weiteren Anforderungen noch zu erfüllen sein werden.

Um die Abstraktion noch weiter zu erhöhen und damit die Programme weiter zu verkürzen, wäre zu erwägen, ob eine noch stärkere Automatisierung zweckmäßig wäre. Mit Hilfe weiterer übergeordneter Klassen könnte eine noch kürzere Formulierung der Programme erreicht werden, wenn Meta-Information über die Tabellenstruktur den Klassen zugänglich wäre. Als Ziel wäre anzustreben, daß zur Definition eines Formulars nur noch die gewünschten Felder angeordnet werden müssen. Alle weiteren Aktionen könnten, wie auch in Access, automatisch ausgeführt werden. Vor der Entwicklung solch einer Erweiterung muß jedoch überprüft werden, ob sie auch für andere Datenbankapplikationen brauchbar wäre.

Kapitel 4

Benutzerhandbuch

Willkommen in der neuen Institutsdatenbank!

Ich möchte Ihnen das neue Verwaltungsprogramm vorstellen und Ihnen erklären, was es alles kann und wie man es verwendet.

Falls Sie noch nicht so häufig mit Anwendungen im Internet gearbeitet haben, darf ich Ihnen das Kapitel 'Erste Schritte' 4.2 und das Anwender-Glossar 4.8 sehr empfehlen. Anschließend wird Sie vielleicht das Kapitel über typische Anwendungsfälle 4.5 interessieren. Es zeigt, wie man die Datenbank sinnvoll verwenden kann, um übliche Aufgaben zu meistern.

Wenn Sie während der Arbeit Tips zu einem bestimmten Formular suchen, verwenden Sie einfach den 'Hilfe'-Link, den Sie auf jeder Seite links im Navigationsbereich finden. Er bringt sie zu einer Hilfeseite, die speziell für jedes Formular Hinweise bereit hält.

Für Administratoren dürfte schließlich das Kapitel über Privilegien 4.4 besonders wichtig sein.

Sie können die Dokumentation natürlich auch von Anfang an zu lesen beginnen; hier die weiteren Kapitel:

Ich beginne mit einem Überblick 4.1, der die Funktionen der Datenbank kurz vorstellt. Die ersten Schritte 4.2 demonstrieren die Verwendung der Formulare, das An- und Abmelden und wie Privilegien funktionieren. Der Abschnitt über Formulare 4.3 gibt Hinweise für das korrekte Ausfüllen von Formularen. Mit der Vergabe von Privilegien beschäftigt sich das nächste Kapitel 4.4. Anschließend wird demonstriert, wie man mit der Datenbank wirklich arbeiten kann 4.5, also zum Beispiel das korrekte Eintragen eines neuen Mitarbeiters. Beim Löschen von Datensätzen kann das nächste Kapitel 4.6 helfen. Zu Ihrer Sicherheit und zur Sicherheit der ganzen Datenbank gibt es auch ein paar Hinweise 4.7. Und falls Ihnen

ein paar der verwendeten Vokabel spanisch vorkommen, kann Ihnen hoffentlich das Glossar mit Erklärungen dienen 4.8.

Nehmen Sie sich bitte ein paar Minuten Zeit für diese Dokumentation – sie kann Ihnen die Arbeit sehr erleichtern!

4.1 Überblick

Zunächst einmal vielen Dank, daß Sie sich entschlossen haben, weiterzulesen!

Die neue Institutsdatenbank soll die Verwaltung unterstützen, indem sie die Daten sammelt, aufbereitet, und auf übersichtliche Art zugänglich macht.

Um Mißverständnissen vorzubeugen, soll gleich zu Beginn deutlich gemacht werden, was diese Datenbank ist – und was nicht: Diese Datenbank dient als 'Behälter' für die Daten von Personen, Räumen, Telefonklappen und weiteren Dingen. Wenn jemand etwas neues in die Datenbank eingibt, sehen auch alle anderen Benutzer den neuen Datensatz – die Datenbank ist also ein gemeinsamer, zentraler Speicher für die Daten, genau wie ein Karteikasten.

Der große Vorteil an einer Datenbank ist jetzt, daß man die Daten nur *ein* mal eingeben muß und sie dann auf unterschiedliche Art wieder ausgeben kann; zum Beispiel wie eine Karteikarte für das Archiv, in Form einer Telefonliste oder als Vorlage für die Gehaltsberechnung. Man erspart sich dadurch Arbeit, weil Daten nur noch einmal eingegeben werden müssen. Wenn die Eingabe aber einen Fehler hat, dann taucht der Fehler überall dort wieder auf, wo dieses Feld verwendet wird - daher ist größere Sorgfalt bei der Eingabe notwendig.

Es soll auch nicht verschwiegen werden, was diese Datenbank *nicht* ist: Sie ist nicht 'intelligent'. Das heißt, die Datenbank ist stur und macht genau das, was der Benutzer vorgibt, ohne mitzudenken. Trotzdem sind ein paar 'Sicherungen' eingebaut, um mögliche Fehler sofort zu erkennen. Es werden zum Beispiel alle eingegebenen Werte überprüft, um offensichtlich falsche Eingaben zu erkennen.

Eine Datenbank speichert Informationen. Aber sie kann nur solche Informationen speichern, für die sie auch ausgelegt ist. Das bedeutet, daß zu Beginn der Entwicklung dieser Datenbank ganz genau festgelegt worden ist, welche Informationen gespeichert werden sollen. Wenn sich neue Anforderungen ergeben, kann die Datenbank natürlich erweitert werden, sie kann aber immer nur ganz genau die Felder (z.B. Vorname, Nachname oder Raumnummer) verarbeiten, die vorgesehen sind.

Diese Datenbank ist tatsächlich nicht viel mehr als eine neue Art von Karteikasten. Die Karten werden aber im Computer gespeichert (man braucht keinen ‘Kasten’ mehr) und es gibt noch ein paar zusätzliche Vorteile: Die Suche nach einer bestimmten Karteikarte kann nach beliebigen Suchbegriffen erfolgen und ist außerdem extrem schnell. Die Informationen werden immer nur ein mal gespeichert und dann untereinander in Beziehung gesetzt (also zum Beispiel: Eine Person gehört zu einer Abteilung; hier gibt es eine Beziehung zwischen Personen und Abteilungen). Da es keine Karteikarten auf Papier gibt, können diese auch nicht verloren gehen und der Karteikasten wird auch nie zu klein.

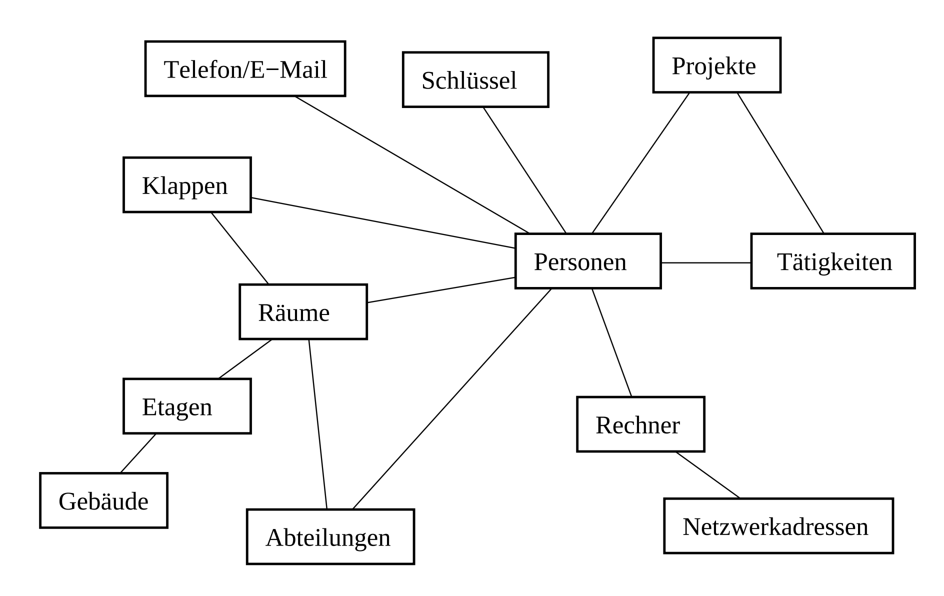


Abbildung 4.1: Übersicht der Institutsdatenbank

Die Abbildung 4.1 zeigt, welche Daten in dieser Datenbank überhaupt gespeichert werden. Die Linien zeigen an, welche Daten miteinander in Beziehung stehen. Jedes der dargestellten Kästchen entspricht einer eigenen Kartei, also einer Sammlung einzelner Datensätze. Jeder einzelne Datensatz (z.B eine Person) besteht aus einzelnen Feldern (Vorname, Nachname. . .) und wird als Formular angezeigt.

4.2 Erste Schritte

Gleich das wichtigste zu allererst: Die Institutsdatenbank ist eine Internet-Anwendung. Deshalb müssen Sie, sobald Sie Änderungen an Daten vornehmen,

die *'Speichern'-Schaltfläche drücken, damit Ihre Änderungen auch tatsächlich gespeichert werden*.
Außerdem: *Verwenden Sie bitte nicht die 'Zurück'-Schaltfläche* Ihres Internetbrowsers. Diese Schaltfläche ist auf normalen Internetseiten sehr hilfreich – bei der Institutsdatenbank führt sie aber nur zu großer Verwirrung, weil Sie beim Zurückgehen Daten wiedersehen, die wahrscheinlich nicht mehr gültig sind.

4.2.1 Starten

Zum Starten der Institutsdatenbank öffnen Sie Ihren bevorzugten Internetbrowser (Netscape, Internet Explorer o.ä.) und geben die Adresse der Datenbank ein. Ihr Administrator wird Ihnen gerne eine Verknüpfung anlegen, die Sie nur anklicken müssen, um die Datenbank zu öffnen. Sie können die Datenbank sofort verwenden. Wenn Sie geschützte Daten nachschlagen oder Änderungen vornehmen möchten, müssen Sie sich anmelden ('login'). Dazu verwenden Sie bitte den 'Login'-Link links oben; sie werden anschließend nach Ihrem Benutzernamen und Kennwort gefragt. Bitte geben Sie Namen und Kennwort genau so ein, wie Sie sie vom Administrator bekommen haben; es kommt auch auf Groß- und Kleinschreibung an.

Wenn Sie sich erfolgreich angemeldet haben, erscheint Ihr Benutzername links oben, andernfalls werden Sie erneut zur Eingabe von Name und Paßwort aufgefordert. Wenn Sie keinen passenden Anmeldenamen haben, können Sie als Benutzername auch 'nobody' angeben und das Paßwortfeld leer lassen; sie haben dann eingeschränkte Rechte.

4.2.2 Navigation

Die Institutsdatenbank ist in mehrere Zweige aufgeteilt, die jeweils zusammengehörige Seiten enthalten. Zu Ihrer Orientierung steht links eine Liste mit allen Zweigen. Der Zweig, den Sie gerade benutzen, ist weiter aufgegliedert und zeigt alle untergeordneten Seiten an. Die Tabreiter am oberen und unteren Rand zeigen hingegen immer nur die benachbarten Formulare an.

Wenn Sie ein Formular öffnen, müssen Sie sich zuerst in einer Liste aussuchen, welchen Datensatz sie genau sehen möchten. Wenn Sie beispielsweise das Personen-Formular öffnen, bekommen Sie zuerst eine Liste mit Namen. Geben Sie ein paar Buchstaben des gesuchten Namens ein und drücken Sie auf 'Suchen'. Dadurch werden in der Liste nur noch die passenden Namen angezeigt. Wenn Sie

den passenden Namen gefunden haben, klicken Sie ihn in der Liste einfach an und Sie werden zu den öffentlichen Daten der Person weitergeleitet. Dieses Prinzip wurde bei allen Seiten umgesetzt, um die Suche nach Datensätzen zu erleichtern.

Beachten Sie auch, daß Sie immer nur jene Verweise im Navigationsbereich links sehen, die Sie aufgrund Ihrer Berechtigungen auch öffnen dürfen. Es ist also normal, daß nach dem Einloggen zusätzliche Links erscheinen.

4.2.3 Ausloggen

Nachdem Sie die Arbeit mit der Datenbank beendet haben, verwenden Sie bitte den 'logout'-Link, um sich abzumelden. Danach können Sie ohne Bedenken das Internetbrowserfenster schließen oder es weiterverwenden, um eine andere Internetseite zu öffnen.

4.2.4 Arbeiten mit Formularen

Wenn Sie einen bestimmten Datensatz geöffnet haben, bekommen Sie den aktuellen Stand der Daten angezeigt. Sie können nun diese Daten ergänzen oder korrigieren. Klicken Sie einfach in das gewünschte Feld und geben Sie neue Werte ein; Sie können auch mit der Maus ziehen (linke Taste drücken, während der Pfeil über dem Text steht und gedrückt halten, während Sie die Maus bewegen) um Text zu markieren. Markierter Text wird automatisch ersetzt, wenn Sie anschließend neue Zeichen eingeben.

Verwenden Sie die Auswahllisten, um Beziehungen zu anderen Datensätzen herzustellen (also zum Beispiel das Sitzzimmer auszuwählen). Beachten Sie, daß Sie verknüpfte Datensätze vielleicht zuerst eingeben müssen, bevor Sie sie in der Auswahlliste anwählen können (wenn es also das Zimmer noch nicht gibt). Um die Suche in langen Listen zu erleichtern, können Sie die Anfangsbuchstaben eingeben (wird nicht in allen Programmen unterstützt).

Wichtig: Drücken Sie unbedingt die 'Speichern'-Schaltfläche, bevor Sie das Formular wieder verlassen! Wenn Sie das vergessen, sind die Änderungen verloren und Sie werden auch nicht gefragt, ob Sie die Änderungen speichern möchten.

4.2.5 Zugriffsrechte

Wenn Sie sich bei der Datenbank anmelden, bekommen Sie durch Ihren Benutzernamen Rechte zuerkannt. Diese Rechte beziehen sich immer auf bestimmte Felder

in der Datenbank und es gibt fünf Stufen der Berechtigung, die aufeinander aufbauen. Wenn Sie zum Beispiel Datensätze neu erstellen dürfen, dann dürfen Sie sie auch lesen und verändern. Jedes Privileg beinhaltet also automatisch alle weniger starken Privilegien.

Privileg	Auswirkung
(keines)	Ohne Privileg dürfen Sie die Daten nicht einmal lesen.
lesen	Sie dürfen die Daten lesen
schreiben	Sie dürfen bestehende Daten bearbeiten
erstellen	Sie dürfen auch neue Datensätze anlegen
löschen	Sie dürfen auch bestehende Datensätze löschen oder wiederherstellen

Tabelle 4.1: Zugriffsrechte

Die Privilegien werden vom Administrator verliehen und gelten jeweils für einen Benutzernamen und für alle Datensätze der jeweiligen Tabelle - unabhängig davon, wer die Datensätze erstellt hat.

4.3 Formulare

Hier ein paar Hinweise zum Benützen der Formulare:

- Geben Sie die Werte in 'normaler' Schreibweise ein. Umlaute und scharfes ß sind erlaubt und sollen verwendet werden. Wenn Sie ungültige Zeichen verwenden, so weist Sie die Datenbank mit einer Fehlermeldung nach dem Speichern darauf hin und Sie müssen die Eingabe korrigieren und erneut speichern.
- Vergessen Sie nicht, nach dem Bearbeiten auf 'Speichern' zu klicken! Achten Sie immer nach dem Speichern darauf, ob eine Fehlermeldung angezeigt wird: Wenn nur *ein* Wert ungültig eingegeben wurde, wird der Speichervorgang abgebrochen und es wird *keiner* der anderen Werte gespeichert!
- Verwenden Sie die 'normale' Groß- und Kleinschreibung. Schreiben Sie nicht in BLOCKBUCHSTABEN!

- Tragen Sie stets nur *einen* Wert in ein Feld ein. Wenn eine Person beispielsweise zwei Kontonummern hat, soll eine davon ausgewählt und eingegeben werden.

- Löschen Sie niemals alle Felder eines Formulars einzeln! Wenn Sie einen Datensatz löschen möchten, lesen Sie bitte im Kapitel 'Löschen und Wiederherstellen' 4.6 weiter und benützen Sie die entsprechende Lösch-Schaltfläche.

- Überschreiben Sie niemals die Felder einer bestehenden Karteikarte mit den Werten eines neuen Datensatzes! Zum Erstellen eines neuen Datensatzes verwenden Sie bitte den entsprechenden Link auf den jeweils vorgelagerten Suchseiten.

- Wenn Sie nicht alle Informationen kennen, die in ein Formular eingetragen werden können, so lassen Sie unbekannte Felder leer. Eingaben so wie 'unbekannt' oder 'wird nachgereicht' sind zumeist nur wenig hilfreich. Wenn die Datenbank das Formular nicht akzeptiert, weil wesentliche Informationen fehlen, müssen Sie diese ermitteln, bevor Sie den Datensatz anlegen können.

4.4 Privilegienvergabe

Das Privilegiensystem der Institutsdatenbank ist sehr flexibel, es muß aber auch mit Vorsicht angewandt werden, um die Kontrolle über die vergebenen Privilegien zu behalten. Das Prinzip ist in Abbildung 4.2 dargestellt. Benutzer können Benutzergruppen zugeteilt werden und Datenbankfelder werden in Feldgruppen eingeordnet. Die Privilegien werden ausschließlich zwischen Benutzergruppen und Feldgruppen vergeben.

Die Abbildung zeigt, daß Benutzer auch mehreren Benutzergruppen angehören können. Während der Benutzer 'Sepp' nur auf die Personendaten zugreifen kann, darf 'Franz' alle dargestellten Felder bearbeiten, weil er der Personalabteilung *und* der Verrechnungsabteilung angehört – für ein bestimmtes Feld gelten also immer die stärksten Privilegien aus allen Gruppenmitgliedschaften.

Das Privilegiensystem ermöglicht auch die vollständige Abtrennung unterschiedlicher Benutzergruppen: Benutzer aus der Personalabteilung, die keinen weiteren Gruppen angehören (in diesem Fall nur 'Sepp'), bekommen die Daten der Tätigkeiten gar nicht zu sehen, weil sie keine Leseberechtigung dafür haben.

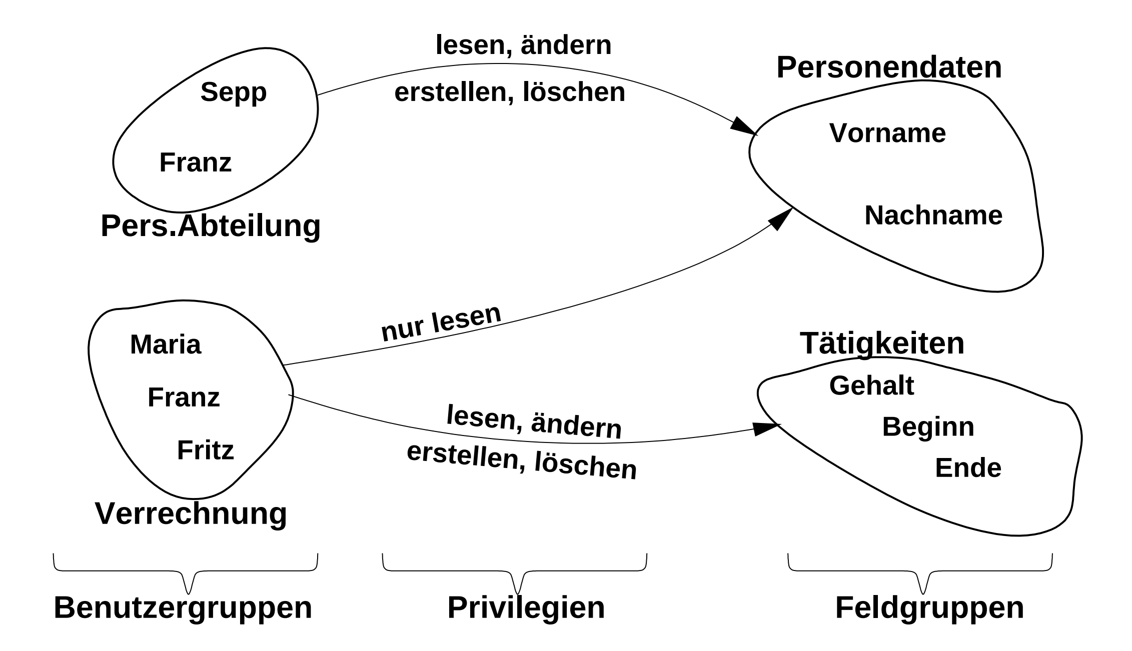


Abbildung 4.2: Prinzip der Privilegienvergabe

Wenn Sie Änderungen an den Privilegien vornehmen, so werden diese Änderungen erst wirksam, wenn sich die betroffenen Benutzer neu anmelden. Beachten Sie auch, daß Sie “sich selbst aussperren” können, wenn Sie der Benutzergruppe, aus der Sie die nötigen Privilegien erhalten, das Privileg zum Ändern der Privilegien entziehen. In diesem Fall muß Ihnen ein anderer privilegierter Benutzer die nötigen Privilegien wieder neu erteilen.

Zur Anwendung des Privilegiensystems finden Sie Beispiele in den ‘typischen Vorgängen’ 4.5.

4.5 Typische Vorgänge

Hier sollen einige typische Arbeitsabläufe dargestellt werden, um die Arbeit mit der Datenbank in der Praxis zu erleichtern. Manche Abläufe können Sie direkt übernehmen, aber für viele andere Aufgaben sind die hier aufgeführten Beispiele nur als Anleitung gedacht.

4.5.1 Hinzufügen eines neuen Datenbank-Benutzers

Wenn Sie einem neuen Benutzer Zugriff auf die Datenbank gewähren wollen, sollten Sie zunächst genau planen, welche Bereiche dem Benutzer zugänglich gemacht werden sollen. Wenn Sie einen neuen Zugang nach dem Muster eines bestehenden einrichten möchten, brauchen Sie nur den neuen Benutzernamen anzulegen und ihn den bereits definierten Benutzergruppen zuzuordnen. Hier sind die einzelnen Schritte der Reihe nach:

- Loggen Sie sich mit einem Benutzernamen ein, der berechtigt ist, Benutzer und Benutzergruppen zu erstellen.
- Öffnen Sie das Benutzerformular (Einstellungen - Privilegien - Benutzer).
- Geben Sie in der ersten Zeile des Formulars einen neuen Benutzernamen und ein Kennwort ein. Den Benutzernamen darf es in der untenstehenden Liste noch nicht geben. Wählen Sie kein allzu einfaches Paßwort mit zumindest drei Zeichen Länge und speichern Sie den Datensatz. Sie haben nun einen neuen Benutzer angelegt, der kein einziges Privileg besitzt.
- Wechseln Sie ins Formular ‘Benutzergruppen’.
- Klappen Sie die gewünschte Benutzergruppe auf (klicken Sie auf den Pfeil links).
- Wählen Sie den neu angelegten Benutzernamen in der Liste aus und klicken Sie auf Hinzufügen. Damit ist der neue Benutzer Mitglied der geöffneten Benutzergruppe geworden und hat alle ihre Berechtigungen.
- Ordnen Sie den Benutzer analog den vorigen beiden Schritten noch weiteren Gruppen zu.
- Mit dem ‘entfernen’-Link können Sie den Benutzer wieder aus der Gruppe austragen.
- Wechseln Sie zurück ins Benutzerformular und öffnen Sie nochmals den neuen Benutzer. Sie sehen alle Privilegien, die der Benutzer nun hat. Kontrollieren Sie mit größter Sorgfalt, ob die Privilegien korrekt vergeben wurden.

Wenn Sie keine passenden Benutzergruppen definiert haben, lesen Sie bitte bei ‘Vergeben von Privilegien’ 4.5.3 weiter. Wenn auch noch keine geeigneten Feldgruppen definiert sind, müssen Sie anhand des nächsten Abschnittes 4.5.2 zuerst welche definieren.

4.5.2 Erstellen von Feldgruppen

Feldgruppen werden verwendet, um mehrere einzelne Felder der Datenbank zu gemeinsamen Einheiten zusammenzufassen, um für alle Felder gemeinsam Privilegien vergeben zu können. Die Aufteilung der Datenbank in geeignete Feldgruppen sollte gut geplant werden, damit man die Übersicht behält. Zum Erstellen einer neuen Feldgruppe gehen Sie so vor:

- Loggen Sie sich mit einem Benutzernamen ein, der berechtigt ist, Feldgruppen zu erstellen.
- Öffnen Sie das Feldgruppenformular (Einstellungen - Privilegien - Feldgruppen).
- Geben Sie in der ersten Zeile des Formulars den Namen der neuen Feldgruppe ein. Der Name darf in der Liste noch nicht vorkommen und sollte möglichst kurz und prägnant beschreiben, welche Daten die Feldgruppe enthalten wird. Klicken Sie auf ‘Neu anlegen’, um die Feldgruppe zu erstellen.
- Sie haben nun eine leere Feldgruppe in der Liste. Klappen Sie sie auf, um Felder hinzuzufügen (klicken Sie auf den Pfeil links).
- Wählen Sie in der Auswahlliste der Reihe nach jene Felder aus, die Sie hinzufügen möchten und klicken Sie zum Hinzufügen auf die entsprechende Schaltfläche. Anstatt alle Felder eines Bereiches einzeln anzuwählen, fügen Sie besser den jeweiligen Eintrag ‘Alle Daten der...’ hinzu.
- Mit den ‘entfernen’-Links können Sie einzelne Felder wieder aus der Gruppe löschen.

Wenn Sie die gewünschten Feldgruppen erstellt haben, können Sie im Privilegienformular die gewünschten Privilegien zwischen Benutzergruppen und Feldgruppen zuordnen. Siehe dazu den nächsten Abschnitt 4.5.3.

4.5.3 Vergeben von Privilegien

In diesem Schritt weisen Sie jeweils einer Benutzergruppe das Recht zu, auf eine vorgegebene Feldgruppe zuzugreifen. Die Benutzergruppe und die Feldgruppe müssen zu diesem Zeitpunkt bereits erstellt sein. Lesen Sie ggf. in den vorigen Abschnitten nach, um Benutzergruppen 4.5.1 und Feldgruppen 4.5.2 zu erstellen.

- Loggen Sie sich mit einem Benutzernamen ein, der berechtigt ist, Privilegien zu vergeben.
- Öffnen Sie das Privilegienformular (Einstellungen - Privilegien).
- Klappen Sie die Benutzergruppe auf, der Sie neue Privilegien geben möchten (klicken Sie auf den Pfeil links). Sie sehen, welche Privilegien die Gruppe momentan besitzt.
- Wählen Sie eine Feldgruppe und das gewünschte Privileg in den Auswahllisten und klicken Sie auf Hinzufügen, um das Privileg zu erteilen.
- Wenn Sie ein bestehendes Privileg ändern möchten, wählen Sie einfach die Feldgruppe in der Auswahlliste aus und geben die neue Privilegienstufe an (so, als würden Sie das Privileg neu zuweisen).
- Um Privilegien wieder zu entziehen, verwenden Sie den entsprechenden Link.
- Wechseln Sie ins Benutzerformular (Einstellungen - Privilegien - Benutzer), nachdem Sie Privilegien bearbeitet haben. Kontrollieren Sie bei einigen Benutzern, ob diese nur jene Privilegien besitzen, die ihnen auch zustehen.

Wenn Sie sich noch nicht näher mit Privilegien beschäftigt haben, sollten Sie die Einführung in Privilegien 4.4 lesen.

4.5.4 Hinzufügen eines neuen Institutsangehörigen

Wenn Sie eine neue Person in die Datenbank eintragen, gibt es zwei Möglichkeiten: Die Person ist wirklich neu oder die Person war bereits am Institut und kehrt nun zurück.

Zum einen: Eingeben einer neuen Person:

- Loggen Sie sich mit einem Benutzernamen ein, der berechtigt ist, Personen anzulegen.

- Öffnen Sie das Personen-Suchformular (Personen).
- Klicken Sie unten auf den Link zum Eingeben einer neuen Person. Das Stammblatt öffnet sich und alle Felder sind leer. (Wenn Sie hier nichts eingeben können, haben Sie nicht die nötigen Privilegien.)
- Füllen Sie das Formular aus. Beachten Sie dazu bitte die Hinweise zum Ausfüllen 4.3!
- Klicken Sie auf Speichern, um die neue Person anzulegen.
- Achten Sie darauf, daß keine Fehlermeldung angezeigt wird! Sie müssen so lange Werte korrigieren und ergänzen, bis keine Fehlermeldung mehr erscheint. Erst dann ist die Person *wirklich* gespeichert.
- Wechseln Sie ins Formular der privaten Daten und geben Sie die Stammdaten der Person ein. Vergessen Sie nicht, auf Speichern zu klicken, wenn Sie fertig sind!
- Sie können nun Privattelefonnummern, E-Mail Adressen, Schlüssel und Telefonklappen zuordnen.
- z.B. E-Mail Adresse: Wechseln Sie ins Formular 'Privatnummern'. Geben Sie im Feld 'Anschluß' die E-Mail Adresse ein. Das Feld 'Text' können Sie leer lassen. Wählen Sie in der Auswahlliste den Anschluß-Typ (E-Mail) aus und klicken Sie das Häkchen an, wenn die Adresse hauptsächlich privat genutzt wird. Klicken Sie auf 'Neu anlegen' und korrigieren Sie die Daten, wenn eine Fehlermeldung angezeigt wird.
- Zum Anlegen von Tätigkeiten lesen Sie bitte im Kapitel 'Personen – Tätigkeiten – Projekte' 4.5.5 weiter.

Zum anderen: Wiedereintritt einer Person (Wiederherstellen): Dieser Fall ist deutlich einfacher, da die Daten bereits gespeichert sind.

- Loggen Sie sich mit einem Benutzernamen ein, der berechtigt ist, Personen *zu löschen*.
- Öffnen Sie das Formular 'Einstellungen' und aktivieren Sie, daß Sie gelöschte Datensätze sehen möchten.
- Öffnen Sie das Personen-Suchformular (Personen).

- Geben Sie einen Teil des Namens im Suchfeld ein und klicken Sie auf Suchen.
- Blättern Sie in der untenstehenden Liste und klicken Sie auf den Namen, wenn Sie ihn gefunden haben.
- Sie werden auf die Karteikarte weitergeleitet; unten wird angezeigt, wann die Person gelöscht wurde.
- Klicken Sie auf ‘Wiederherstellen’. Wenn das Wiederherstellen nicht gelingt, stellen ordnen Sie zuerst der gelöschten Person eine andere - nicht gelöschte - Abteilung, Sitzzimmer und Faxklappe zu, speichern Sie und versuchen Sie es erneut.
- Kontrollieren Sie in allen Formularen, ob die eingegebenen Informationen noch korrekt sind.
- Zum Anlegen von Tätigkeiten lesen Sie bitte im Kapitel ‘Personen – Tätigkeiten – Projekte’ 4.5.5 weiter.

4.5.5 Personen – Tätigkeiten – Projekte

Personen üben Tätigkeiten aus, zum Beispiel könnte eine Person gleichzeitig Diplomand und Lehrbeauftragter sein. Personen nehmen auch an Projekten teil. Hier muß nun aber unterschieden werden: Wenn der Diplomand X an einem Projekt teilnimmt, bekommt er keine Bezahlung - derselbe X wird aber sehr wohl entlohnt, wenn er in seiner Funktion als Angestellter an demselben Projekt mitarbeitet. Deshalb nimmt in der Institutsdatenbank sozusagen nicht eine Person an einem Projekt teil, sondern eine spezielle Tätigkeit einer Person.

Um für eine Person eine Tätigkeit einzugeben, muß die Person bereits in der Datenbank vorhanden sein. Wenn das nicht der Fall ist, lesen Sie bitte im Kapitel ‘Eingeben von Institutsangehörigen’ 4.5.4 weiter. Wenn die Person also bereits eingegeben ist, beachten Sie folgende Schritte, um eine Tätigkeit anzulegen:

- Loggen Sie sich mit einem Benutzernamen ein, der berechtigt ist, Tätigkeiten zu erstellen.
- Öffnen Sie das Suchformular der Personen und geben Sie einen Teil des Namens ein, um die richtige Person schneller zu finden. Klicken Sie auf ‘Suchen’ und wählen Sie den Namen in der angezeigten Liste; Sie werden auf die Karteikarte der Person weitergeleitet.

- Wechseln Sie in das Tätigkeiten-Formular.
- In der Auswahlliste ganz oben sehen Sie alle Tätigkeiten, die diese Person gerade ausführt. Zum Eingeben einer neuen wählen Sie ‘Neue Tätigkeit’. Je nach Art der Tätigkeit wählen Sie bitte als nächstes den richtigen Tätigkeitstyp in der zweiten Auswahlliste. Dadurch erhalten Sie die passenden Eingabefelder für diesen Tätigkeitstyp angezeigt.
- Füllen Sie nun alle Felder aus
- Speichern Sie die Tätigkeit. Wenn eine Fehlermeldung angezeigt wird, korrigieren Sie die Eingabe und ergänzen Sie benötigte Felder. Erst, wenn keine Fehlermeldung nach dem Speichern angezeigt wird, ist die Tätigkeit wirklich gespeichert.

Um die Tätigkeit der Person an einem Projekt teilnehmen zu lassen, folgen Sie den nächsten Schritten:

- Falls Sie es noch nicht getan haben: Loggen Sie sich mit einem ausreichend privilegierten Benutzernamen ein und öffnen Sie die Karteikarte der richtigen Person.
- Wechseln Sie ins Projekteformular.
- Wählen Sie zuerst aus, welche Tätigkeit an dem Projekt teilnehmen soll. Davon hängt es ab, ob aus der Mitarbeit eine Bezahlung resultiert.
- Wählen Sie das Projekt in der zweiten Auswahlliste und geben Sie ein, ab wann die Teilnahme erfolgt und ob daraus tatsächlich eine Bezahlung resultiert.
- Speichern Sie die Projektteilnahme und überprüfen Sie, daß keine Fehlermeldung angezeigt wird.
- Wechseln Sie ins Funktionen-Formular. Hier wird die neue Tätigkeit bzw. die Teilnahme am Projekt angezeigt.

4.5.6 Austritt eines Institutsangehörigen

Wenn Personen das Institut wieder verlassen, sollten sie auch aus der Datenbank gelöscht werden, damit sie nicht mehr auf Telefonlisten o.ä. erscheinen. Bevor eine Person aber gelöscht werden kann, müssen alle ‘Beziehungen’ gelöst werden: Die Person darf keine Tätigkeiten mehr ausüben, sie muß die Verantwortlichkeit für Abteilungen, Räume und Computer an andere Personen übergeben haben und darf keine Projekte mehr leiten.

Um eine Person zu löschen, beachten Sie die folgenden Schritte:

- Loggen Sie sich mit einem Benutzernamen ein, der berechtigt ist, Personen zu löschen.
- Öffnen Sie das Suchformular der Personen und geben Sie einen Teil des Namens ein, um die richtige Person schneller zu finden. Klicken Sie auf ‘Suchen’ und wählen Sie den Namen in der angezeigten Liste; Sie werden auf die Karteikarte der Person weitergeleitet.
- Klicken Sie auf die Löschen-Schaltfläche im Formular ‘Öffentliche Daten’.
- Wenn das Löschen funktioniert hat, sehen Sie nun das Wort ‘gelöscht’ und am unteren Ende der Karteikarte wird das Löschdatum angezeigt. Viel wahrscheinlicher ist aber der andere Fall...
- Wenn das Löschen nicht möglich war, bekommen Sie eine Liste mit den Hinderungsgründen, also allen bestehenden Beziehungen. Klicken Sie der Reihe nach auf jeden der Links, um die konkrete Beziehung anzuzeigen.
- Wenn die Person noch Tätigkeiten ausübt, löschen Sie die Tätigkeiten. Wenn die Person für etwas verantwortlich ist (Computer, Räume, Projekt- oder Institutsleitung) dann wählen Sie an den entsprechenden Stellen einen neuen Verantwortlichen aus, um die Beziehung zu lösen.
- Wenn alle Beziehungen gelöst und alle Verantwortlichkeiten abgegeben sind, können Sie die Person tatsächlich löschen.

4.5.7 Anlegen eines neuen Raumes

Stellvertretend für viele andere Formulare wird hier gezeigt, wie man einen neuen Raum eingibt. Die Formulare funktionieren alle sehr ähnlich, deshalb gilt das gesagte auch für andere Fälle.

- Loggen Sie sich mit einem Benutzernamen ein, der berechtigt ist, Räume anzulegen.
- Öffnen Sie das Räume-Suchformular (Räume).
- Klicken Sie unten auf den Link zum Eingeben eines neuen Raumes. Die Karteikarte öffnet sich und alle Felder sind leer. (Wenn Sie hier nichts eingeben können, haben Sie nicht die nötigen Privilegien.)
- Füllen Sie das Formular aus. Beachten Sie bitte dazu die Hinweise zum Ausfüllen 4.3!
- Klicken Sie auf Speichern, um den neuen Raum anzulegen.
- Achten Sie darauf, daß keine Fehlermeldung angezeigt wird! Sie müssen so lange Werte korrigieren und ergänzen, bis keine Fehlermeldung mehr erscheint. Erst dann ist der Raum *wirklich* gespeichert.
- Sie können nun den Raum mit anderen Datensätzen in Verbindung bringen: Sie können ins Klappenformular wechseln und bei einer Klappe den soeben erstellten Raum als Standort wählen. Sie können auch ins Personenformular wechseln und bei den öffentlichen Daten einer Person den Raum als Sitzzimmer angeben. . .

Ab sofort erscheint der neu eingegebene Raum in allen Auswahllisten, wo Räume angezeigt werden, und Sie können ihn auswählen.

4.5.8 Eine Suchabfrage formulieren

Das Suchformular ermöglicht es Ihnen, der Datenbank eine Frage zu stellen. Sie geben an, welche Felder Sie sehen möchten und welche Kriterien die Datensätze erfüllen müssen, damit sie angezeigt werden. Wenn Sie einfach nur Datensätze auffinden möchten, dann ist das Volltext-Suchformular im Service-Bereich besser geeignet. Geben Sie einfach einen Teil eines Namens, eines Projektthemas, einer Telefonklappe usw. ein. Verwenden Sie die in den Treffern angezeigten Links, um die jeweilige Karteikarte zu öffnen.

Zum Erstellen einer neuen Abfrage gehen Sie so vor:

- Öffnen Sie das Suchformular (Service-Center - Such-Abfrage)

- Wählen Sie bei ‘Suchen nach’ aus, welchen Bereich der Datenbank Sie durchsuchen möchten. Dadurch entscheiden Sie, welche Felder Sie in der Abfrage verwenden können.
- Wählen Sie jetzt in der mehrzeiligen Auswahlliste die Felder aus, die Sie als Ergebnis anzeigen möchten. Sie können mehrere Felder auswählen, indem Sie die Großschreibe- oder die Steuerungstaste halten, während Sie die einzelnen Feldnamen anklicken.
- Zum Überprüfen Ihrer Abfrage können Sie bereits jetzt mit der ‘Suchen’-Schaltfläche das Ergebnis anzeigen lassen. Sie bekommen eine Liste mit allen ausgewählten Feldern.
- Um die angezeigten Datensätze einzuschränken, geben Sie ein Kriterium ein: Wählen Sie dazu das Feld, das Sie zum Einschränken verwenden möchten in der Kriterienzeile aus. Wählen Sie weiters eine Vergleichsfunktion (kleiner, größer, wie) und geben Sie im Textfeld einen Vergleichswert an. Sie können auch mehrere Kriterien mit ‘und’ bzw. ‘oder’ verknüpfen. Überprüfen Sie zwischendurch, ob die Ergebnismenge Ihren Erwartungen entspricht.
- Sie können auch Felder auswählen, nach denen sortiert werden soll.
- Speichern Sie einmal erstellte Abfragen ab, um sie später wiederzuverwenden. Geben Sie einen aussagekräftigen Namen an und klicken Sie auf ‘Speichern’.

4.6 Löschen und Wiederherstellen

Zum Löschen von Datensätzen brauchen Sie das entsprechende Privileg. Erst mit diesem Privileg bekommen Sie die ‘Löschen’-Schaltfläche.

Wenn ein Datensatz ungültig geworden ist (die Person ist ausgeschieden, der Raum wurde an ein anderes Institut abgegeben usw.), ist der Datensatz zu löschen. **Wichtig:** Löschen Sie niemals die Werte, die in den Textfeldern eingetragen sind! Wenn ein Datensatz gelöscht werden soll, dann ist das mit der ‘Löschen’-Schaltfläche zu erledigen.

Bevor Sie einen Datensatz löschen können, darf er mit keinen anderen Datensätzen mehr in Beziehung stehen. Sie können eine Person zum Beispiel erst

dann löschen, wenn sie keine Projekte mehr leitet und keine Klappen benutzt usw. Wenn noch Beziehungen bestehen, bekommen Sie nach dem Lösch-Versuch eine Liste mit den bestehenden Beziehungen präsentiert. Sie müssen alle Beziehungen lösen (indem Sie die entsprechenden Datensätze löschen), bevor Sie den eigentlichen Datensatz löschen können.

Nachdem ein Datensatz gelöscht wurde, fällt er aus allen Auswahllisten heraus und wird nur dann noch angezeigt, wenn Sie bei den 'Einstellungen' die entsprechende Funktion aktivieren (sie müssen auch die nötigen Privilegien besitzen). Im Formular werden die Wiederherstellen-Schaltfläche und das Löschdatum angezeigt.

Wenn ein Datensatz später wieder gebraucht wird (weil etwa die Person wieder ans Institut zurückkehrt), so kann der Datensatz wiederhergestellt werden. Beim Wiederherstellen sind ähnliche Abhängigkeiten wie beim Löschen (nur in umgekehrter Reihenfolge) zu berücksichtigen: Ein Datensatz kann nur dann wiederhergestellt werden, wenn kein Datensatz, von dem er abhängig ist (z.B. sein Sitzzimmer, seine Abteilung) gelöscht ist.

4.7 Sicherheit

Zur Sicherheit für die Datenbank und zur Sicherheit Ihres Computers hier ein paar Tips, die Sie beachten sollten:

- Geben Sie Ihr Paßwort nicht weiter und kleben Sie es nicht auf Ihren Computer. Nein, auch nicht auf Ihren Monitor.
- Lassen Sie Ihren Computer nie alleine, wenn die Datenbank geöffnet ist. Am besten wäre es, den Computer durch eine Paßwort-Sperre zu schützen, wenn Sie den Arbeitsplatz verlassen – auch wenn es nur für kurze Zeit ist.
- Klicken Sie auf 'logout', bevor Sie die Datenbank verlassen. Dadurch kann niemand in Ihrem Namen Schaden anrichten.
- Schützen Sie sich vor Viren: Öffnen Sie keine E-Mails, die suspekt erscheinen (englischsprachiger Betreff; Absender, die Sie nicht kennen; 'verlockende' Angebote) und bleiben Sie 'obskuren' Internetseiten fern.
- Verwenden Sie stets die (zweit-)aktuellste Browsersoftware, um Programmfehlern und Sicherheitslücken zu entgehen.

4.8 Anwender-Glossar

Hier werden Begriffe erklärt, ohne die die Beschreibung der Institutsdatenbank kaum möglich wäre…

Begriff	Erklärung
Abfrage	Mit einer Abfrage kann man Fragen an die Datenbank richten. Zu einer Abfrage muß man angeben, welche Felder (Spalten der Tabellen) angezeigt werden sollen und welche Kriterien Datensätze (Zeilen der Tabellen) erfüllen müssen, damit sie angezeigt werden. Eine Frage wäre beispielsweise ‘Zeige mir aus der Personen-Tabelle die Spalten Vorname, Nachname und Ort von jenen Datensätzen, bei denen der Geburtstag im März liegt’. Sie können selbst Abfragen erstellen; benutzen Sie dazu das Suchabfragenformular in der Sparte ‘Service’.
Benutzername	Der Name, mit dem Sie sich ‘ausweisen‘, wenn Sie die Datenbank benützen. Anhand Ihres Benutzernamens wird entschieden, welche Rechte (Privilegien) sie besitzen – welche Aktionen Sie also ausführen dürfen.
Browser	Programm, mit dem Sie Internetseiten öffnen. Sehr weit verbreitet sind Netscape Navigator und Internet Explorer.
Datenbank	Eine Datenbank ist eine Sammlung von Daten, die alle an ein Datenbankschema angepaßt sind. Das bedeutet, daß – wie auf einem Antragsformular – nur ganz bestimmte und zuvor festgelegte Aspekte behandelt werden. Eine Datenbank besteht aus Tabellen und diese wiederum aus einzelnen Datensätzen.
Datensatz	Zeile in einer Tabelle. Jeder Datensatz ist eine eigenständige Einheit und bezieht sich meistens auf etwas ganz konkretes, worüber Daten gespeichert werden sollen (ein Raum, eine Telefonklappe, ein Projekt oder eine Person).

Feld	Ein einzelner Wert (zumeist ein Text oder eine Zahl). Jeder Datensatz besteht aus Feldern, wie sie auch auf einer Karteikarte zu finden wären (Vorname, Nachname, Adresse...). Erst alle Felder zusammen ergeben den kompletten Datensatz. Felder können auch dazu verwendet werden, um auf andere Datensätze zu verweisen: Zu jeder Person wird das Sitzzimmer gespeichert. Das dazu verwendete Feld beinhaltet aber nur einen Verweis auf den jeweiligen Zimmer-Datensatz und genaue Informationen über das Sitzzimmer (z.B. die Ausmaße) findet man, wenn man in der Zimmer-Tabelle nachschlägt.
Formular	Wie in der realen Welt auch ist ein Formular zum Eingeben von Daten gedacht. Wenn in einem Formular genau *ein* Datensatz angezeigt wird, dann kann man das Formular wie eine Karteikarte sehen. Man kann sich auch ein bereits ausgefülltes Formular wieder anzeigen lassen, um Informationen nachzuschlagen.
Paßwort	Ein paar Zeichen, die nur der ‘echte’ Benutzer wissen darf, um sich auszuweisen. Paßwörter müssen unbedingt geheim gehalten werden und dürfen nicht zu einfach gewählt sein. Beachten Sie auch, daß alle Aktionen, die Sie ausführen, aufgezeichnet werden können. Wenn Sie also Ihr Paßwort weitergeben, übernehmen Sie die Verantwortung für alles was damit angerichtet wird.
Server	Ein Server ist (vereinfacht) ein Computer, der an das Netzwerk (zumeist das Internet) angeschlossen ist, und der ein “Service” anbietet. Zum Beispiel ein Web-Server: Er bietet Internetseiten zum Herunterladen und Ansehen an. Oder der Server der Institutsdatenbank: Er bietet die gespeicherten Daten und ein Programm zum Bearbeiten der Daten an.

Tabelle	Eine Tabelle ist eine Ansammlung von gleichartigen Datensätzen. Sie besteht aus Spalten (auch Felder genannt), deren Name und Bedeutung im Entwurf der Datenbank festgelegt wird, und aus Zeilen, den Datensätzen. Jede einzelne Person wird also in einer eigenen Zeile in der ‘Personen’-Tabelle gespeichert; die Spalten heißen zum Beispiel ‘Nachname’, ‘Adresse’ oder ‘Geburtsdatum’.

Anhang A

Abkürzungen und Begriffe

Die folgenden Begriffserklärungen beziehen sich hauptsächlich auf das Beispiel einer http-Session in Kapitel 2.3.2.

ARP	Address Resolution Protocol. Ein Protokoll, mit dem zu einer bekannten IP-Adresse (Layer 3) die zugehörige Hardware-Adresse (MAC-Adresse, Layer 2) ermittelt werden kann, um den zugehörigen Rechner direkt ansprechen zu können.
Client	In einer Client-Server-Beziehung ist der Client jener Partner, der einen Dienst von einem Server abruft. Es kann sich dabei um unterschiedlichste Arten von Diensten handeln; etwa eine Datenbank-Auskunft oder der Abruf einer html-Seite.
IP-Adresse	Internet-Adresse. Mit dem Internet verbundene Rechner können über diese (zumindest temporär) weltweit eindeutige Adresse angesprochen werden. Im Gegensatz zur MAC-Adresse ist die IP-Adresse strukturiert (d.h. routbar). IP-Datenpakete können sehr effizient an ihr Ziel weitergeleitet werden, auch wenn der Standort des Rechners zu Beginn der Übertragung unbekannt ist.
MAC-Adresse	Global eindeutige Hardware-Adresse, die in jeder Netzwerkkarte fest einprogrammiert ist. Nur über diese Adresse kann das Interface tatsächlich angesprochen werden.

Port	Die Portnummer bestimmt, welcher Server-Prozeß ankommende Daten bekommen soll. Für Web-Server wird üblicherweise die Portnummer 80 verwendet.
Server	Server stellen Dienste bereit, die von Clients abgerufen werden können. Von einem Webserver kann man beispielsweise über das htt-Protokoll Internetseiten abrufen. Selbstverständlich kann auch ein Server selbst zum Client werden, wenn er zum Aufbau einer Internetseite einen Datenbankserver konsultiert.
TCP-Session	Eine virtuelle Netzwerkverbindung zwischen zwei Rechnern über das Internet. Sie ist eindeutig gekennzeichnet durch die beiden IP-Adressen samt Portnummern. Die beiden Rechner sind prinzipiell gleichberechtigt (jeder der beiden könnte unaufgefordert Daten senden), der Ablauf wird aber meist durch höhere Protokolle (etwa http) gesteuert.

Anhang B

Löschabhängigkeiten

Die folgende Tabelle zeigt die Abhängigkeiten, die beachtet werden müssen, bevor ein Datensatz gelöscht oder wiederhergestellt werden darf.

Die Abhängigkeiten lassen sich direkt aus dem EER-Diagramm (Abb. 2.1) ableiten. Beim Löschen eines Datensatzes dürfen keine nicht gelöschten Datensätze auf keiner der n-Seiten seiner Beziehungen vorhanden sein und beim Wiederherstellen darf auf keiner der 1-Seiten seiner Beziehungen der Datensatz gelöscht sein. Anschaulich betrachtet zeigen die schwarzen Hälften der Beziehungssymbole (als Pfeil betrachtet) in Richtung der Löschabhängigkeit und die weißen Hälften weisen in Richtung der Wiederherstellungsabhängigkeit.

Wenn beispielsweise eine Etage gelöscht werden soll, so darf sie ausschließlich gelöschte Räume enthalten. Wird eine Etage wiederhergestellt, so muß überprüft werden, ob das Gebäude nicht gelöscht ist.

Tabelle	Löschabhängigkeit	Löschweitergabe	Wiederherstellungsabhängigkeit
Personen	Rechner, Taetigkeiten (eigene sowie betreute), Projekte (Leitung), Raeume (Verantwortklichkeit), Organisationseinheiten (1x Leitg und 3x Stellv), Schluessel	Anschluesse, KlapPersZuord	Klappen (nur Fax-Klappen), Raeume (Sitzzimmer der Pers.), OrganisationsEinheiten (Angehörigkeit)
Anschluesse	keine	keine	AnschlussTypen, Personen
AnschlussTypen	Anschluesse, Klappen	keine	keine
Klappen	KlapPersZuord, Personen (nur FaxKlappen)	keine	AnschlussTypen, Raeume
KlapPersZuord	keine	keine	Klappen, Personen
Raeume	Klappen, Personen (Sitzzimmer)	keine	Personen (Verantwortlicher), Etagen, RaumNutzungen, OrganisationsEinheiten
Etagen	Raeume	keine	Gebaeude
Gebaeude	Etagen	keine	keine
RaumNutzungen	Raeume	keine	keine
OrganisationsEinheiten	OrganisationsEinheiten (nur untergeordnete), Raeume, Personen (Institutsangehörige)	keine	Personen (1x Leiter und 3x Stellv), OrganisationsEinheiten (die übergeordnete), OrganisationsTypen
OrganisationsTypen	OrganisationsEinheiten	keine	keine
Rechner	IPAdressen	keine	Personen (Verantwortlicher)
IPAdressen	keine	keine	Rechner

Tabelle	Löschabhängigkeit	Löschweitergabe	Wiederherstellungsabhängigkeit
Schluessel	keine	keine	Personen, SchluesselTypen
SchluesselTypen	Schluessel	keine	keine
Projekte	ProjektTeilnahmen (nur aktive)	keine	Personen (Projektleiter), AuftraggeberTypen
AuftraggeberTypen	Projekte	keine	keine
Refundierungen	Taetigkeiten	keine	keine
FeldSetups	TaetTypen	keine	keine
TaetTypen	Taetigkeiten	keine	FeldSetups
Taetigkeiten	ProjektTeilnahmen (nur aktive)	keine	Personen (eigene und betreuende), Refundierungen, TaetTypen
ProjektTeilnahmen	keine	keine	Projekte, Taetigkeiten
Attribute	keine	keine	keine
FeldGruppen	keine	FelderInGruppen, Privilegien	n.a.
FelderInGruppen	keine	keine	n.a.
Privilegien	keine	keine	n.a.
user	keine	user_group, SessionPrivilegien, u. Sessions	n.a.
user_group	keine	keine	n.a.
UserGruppen	keine	user_group, Privilegien	n.a.

Löschabhängigkeiten und Wiederherstellungsabhängigkeiten

Anhang C

Typische Fehlerquellen

Während der Entwicklung sind dem Autor einige Fehler passiert, die jeweils ein sogenanntes 'Aha'-Erlebnis bewirkt haben, als ihre Ursachen endlich entdeckt waren. In der Hoffnung, daß sie für Entwickler von Internet-Applikationen hilfreich seien, sollen sie hier ohne Anspruch auf Vollständigkeit kurz aufgezählt werden.

Browser

- Nennen Sie einen html-Button niemals 'submit'! Sonst funktioniert die Methode `this.form.submit()` nicht mehr.
- Manche Benutzer verwenden die Möglichkeit, Formulare abzuschicken, indem sie den Submit-Button mit der Leertaste betätigen, wenn er den Fokus besitzt. Damit auch in diesem Fall und in Verbindung mit dem Internet Explorer die gewünschten JavaScript Routinen vor dem Submit ausgeführt werden, darf das Ereignis `onClick()` nicht verwendet werden; statt dessen muß immer `onSubmit()` verwendet werden.

Datenbank

- In SQL String-Verkettung verwendet man besser die MySQL-Funktion `CONCAT_WS` anstatt `CONCAT`. `CONCAT_WS` ignoriert NULL-Werte in einzelnen Feldern, bei `CONCAT` wird der gesamte String NULL.
- Bei Verwendung von Verknüpfungsausdrücken der Form `FROM TabelleX LEFT JOIN TabelleY ON ( {Kriterien} )` dürfen bei den 'Kriterien'

nur jene Felder verwendet werden, die auch an der Verknüpfung beteiligt sind (also Schlüsselfelder sind). Weitere Bedingungen müssen im `WHERE`-Abschnitt untergebracht werden.

Datenbank

- JavaScripts müssen unter exakter Einhaltung von Zeichensetzung in html eingebettet werden:

  ```
  <script type='text/javascript' language='JavaScript'>
  <!--
    /* JavaScript-Code */
  //-->
  </script>
  ```

  Es kommt dabei auch auf die Zeilenumbrüche vor und nach den html-Kommentaren an. Diese Information stammt aus SelfHTML (siehe auch: `selfhtml8/javascript/intro.htm#javascriptbereiche`).

- JavaScript: Die Methode zur Zeichenketten-Ersetzung `String.replace()` mit der Option 'g' (wie 'global') funktioniert nicht mit Konqueror 2.2.1 (er ersetzt nur das erste Vorkommen). Man muß sich deshalb eine eigene Funktion aufbauen.

- Konqueror 2.2.1 ignoriert auch die Option 'i' (wie case Insensitive): er sucht bei JavaScript-Regulären Ausdrücken immer case sensitive

php

- Durchlaufen von Feldern eines Array: Vor einem
  "`while (list($key, $value) = each($feld)) ...` "
  darf niemals auf das
  "`reset($feld)`"
  vergessen werden. Der Fehler zeigt sich durch seltsames Verhalten: Es wird nur das erste oder letzte(?) Element bearbeitet. - So, als wäre das Feld abgeschnitten worden.

- Wenn eine Variable (zumeist eine Formularvariable) einen Wert haben *müßte* und trotzdem im Programm *keinen* Wert hat, dann ist sie wahrscheinlich nicht 'globalisiert' worden. Bei der Institutsdatenbank läuft der gesamte

Code im Kontext einer Klassenmethode. Deshalb müssen alle globalen Variablen (am besten gleich zu Beginn der Methode) durch
`global $variable;`
erst zugänglich gemacht werden.

- Fragezeichen-Doppelpunkt-Operatoren, die in php-Stringverkettungen eingebaut sind, sollen in Klammern eingefaßt werden, da sonst der Ergebnisstring seltsam abgeschnitten wird (php 4.04pl1). So sollte es gemacht werden:
  `cout("Anzahl: $count Person" . ($count==1 ? '' : 'en'));`

- Leerzeilen in php-Dateien vor dem öffnenden oder nach dem schließenden php-Tag können seltsame Fehlermeldungen erzeugen:
  "Warning: Cannot add header information - headers already sent by ... "
  In der angegebenen Datei ist wahrscheinlich am Ende noch eine Leerzeile vorhanden. Es darf aber keine Ausgabe erfolgen, bevor die Session-Klasse den Content-Type ausgegeben hat.

Noch ein kleiner Tip, der unter Linux die Arbeit beim Ausbessern in mehreren Dateien sehr erleichtern kann: Verwenden Sie `grep` zum Suchen von Zeichenfolgen in den Quelldateien. Beispiel: `grep "?.*:" *` sucht nach Zeilen, in denen ein Fragezeichen und danach irgendwo ein Doppelpunkt vorkommt. Siehe dazu auch die Hilfe-Seiten zu `grep` und regulären Ausdrücken (Quellen-Hinweise bei `my_form->add_text_element()`, `local.inc`).

Literaturverzeichnis

[1] TCP/IP Tutorial and Technical Overview
Muhrhammer, Atakan, Bretz, Pugh, Suzuki, Wood
Online unter http://www.redbooks.ibm.com

[2] Ich bin so frei
MySQL und PostgreSQL im Vergleich
Jürgen Mischke
iX, 1/2002, Seite 50 bis 65
Verlag Heinz Heise

[3] SelfHTML 8.0
Stefan Münz
Online unter http://selfhtml.teamone.de

[4] JavaScript Guide – client side JavaScript
Netscape
http://developer.netscape.com/docs/manuals/js/client/jsguide/ClientGuideJS13.pdf

[5] JavaScript Reference – client side and server side JavaScript
Netscape
http://developer.netscape.com/docs/manuals/js/client/jsref/ClientReferenceJS13.pdf

[6] The MySQL Manual
MySQL AB
Online unter www.mysql.com/documentation

[7] PHP-HOWTO
Alsvoor Vasudevan
Online unter http://metalab.unc.edu/LDP/HOWTO/PHP-HOWTO.html

[8] The PHP Manual
Stig Sæther Bakken, Egon Schmid
Online unter http://www.php.net/download-docs.php

[9] Deutschsprachige php-FAQ
Zusammenfassung der Newsgroup de.comp.lang.php
Kristian Köhntopp
Online unter http://dclp-faq.de/faq-html.tar.gz

[10] Web Usability, Das Prinzip des Vertrauens
Martina Mannhartsberger, Sabine Musil
Galileo Design
ISBN 3-89842-187-2

[11] The World Wide Web Consortium
Standards für das Internet, Validierungsservice Online unter www.w3.org

[12] Datenbanksysteme
Skriptum zur Vorlesung 181.038
Gerald Pfeifer, Michael Schrefl, Katrin Seyr, Markus Stumptner
Institut für Datenbanksysteme und Artificial Intelligence, TU Wien

[13] Ausgewählte Kapitel der Informatik, Teil 1 Datenbanken
Skriptum zur Vorlesung 384.009
Karl Michael Göschka
Institut für Computertechnik, TU Wien

[14] Datenbanken: Konzepte und Sprachen
Heuer, Saake
Thomson Publishing, Bonn

[15] PHPLIB Dokumentation
Boris Erdmann, Kristian Köhntopp, Sascha Schumann
Online unter http://phplib.sourceforge.net

[16] Merkblatt für den Aufbau wissenschaftlicher Arbeiten
Karl Michael Göschka
Institut für Computertechnik, TU Wien
Online unter http://www.ict.tuwien.ac.at

[17] The Not So Short Introduction to LaTeX 2ε
Or LaTeX 2ε in 95 minutes
Tobias Oetiker, Hubert Partl, Irene Hyna und Elisabeth Schlegl
erhältlich Online bei CTAN, http://www.ctan.org

[18] `mod_ssl` User Manual
Ralf S. Engelschall
Online unter: http://www.modssl.org
oder als Teil der Apache-Dokumentation unter
/usr/share/doc/packages/apache/manual/mod/mod_ssl/index.html